Enginyeria en Informàtica

Projecte Final de Carrera

Mètode espectral
per a l'aprenentatge de màquines d'estats

Memòria final

Albert Santiago Boíl

3 de juny de 2013

Índex

1 Introducció

1.1 Origen i antecedents del projecte

1.1.1 Sobre el projecte

El grup LARCA (Laboratori d'Algorísmia Relacional, Complexitat i Aprenentatge) és un grup de recerca de projecció internacional en temes d'aprenentatge artificial, mineria de dades, recuperació d'informació, lingüística matemàtica i anàlisi de xarxes complexes. Particularment, en l'àmbit de la recerca aplicada i la transferència de la tecnologia, col·labora amb empreses i institucions per tal d'aplicar els seus coneixements als problemes del món real.

Entre els mesos de novembre de 2012 i abril del 2013, he gaudit d'una beca d'iniciació a la recerca dins del grup LARCA, i un dels objectius d'aquesta beca era la realització d'aquest projecte.

L'obtenció de models que expliquin dades observades és un dels propòsits sovint més perseguits en l'àrea de l'aprenentatge automàtic. Aconseguir un bon model de les observacions permet respondre moltes qüestions referents a allò que se'n pot esperar en el futur. Per posar un exemple, una bona modelització de l'ús del transport públic en una ciutat permetrà una millor adequació d'aquest a les necessitats dels usuaris.

En aquest àmbit de l'aprenentatge automàtic, les màquines d'estats finits (en destaquem els autòmats i els hidden Markov models) formen una classe de models de referència molt important, en la qual s'ha centrat l'esforç de molts investigadors. Precisament, una de les fites del grup durant els últims anys és el desenvolupament, en col·laboració amb investigadors d'altres universitats, del *mètode espectral per a l'aprenentatge de d'autòmats finits*, que proporciona una millora substancial en l'*eficiència* de l'aprenentatge de màquines d'estats respecte d'altres mètodes desenvolupats fins al moment. Això sense deixar de banda la correctesa dels models obtinguts: el mètode espectral té garanties teòriques de bon funcionament.

L'objectiu últim del projecte seria oferir a la comunitat científica una bona implementació del mètode espectral, correctament documentada i integrada en una plataforma d'ús habitual en l'àmbit de la inferència estadística. A data d'avui, no existeix a la comunitat un producte d'aquestes característiques.

És per aquest motiu que se'm va proposar realitzar el projecte en R, un llenguatge de programació lliure i gratuït i utilitzat àmpliament per al desenvolupament de programes estadístics i d'anàlisi de dades. D'altra banda, un dels objectius de LARCA és donar-se a conèixer a investigadors d'arreu del món en mineria de dades i *machine learning*, i aquest producte podria contribuir molt positivament en aquest sentit.

Una gran fita que es podria assolir amb aquest projecte seria la publicació del paquet de R desenvolupat al CRAN *(Comprehensive R Archive Network)*, repositori oficial de paquets del llenguatge R. És per això que se n'hauria de redactar la documentació en anglès.

1.1.2 Sobre mi

Sóc estudiant de la doble titulació en Llicenciatura en Matemàtiques i Enginyeria en Informàtica al Centre de Formació Interdisciplinària Superior (CFIS) de la UPC. Aquest projecte es la culminació de cinc anys d'intens estudi i dedicació a la universitat.

Si bé curricularment el Projecte de Final de Carrera és, segons es defineix a la normativa, un *exercici de reválida per a l'Enginyeria Informàtica*, la meva intenció és que aquest Projecte serveixi com a punt i final d'ambdues titulacions.

És per això que el tema escollit té un forta component matemàtica i s'ha posat èmfasi en alguns dels aspectes més matemàtics del projecte. Hi ha contribuït notablement el fet que el director del projecte, en Borja de Balle, va estudiar també una doble titulació, en matemàtiques i enginyeria de telecomunicacions.

1.2 Objectius del projecte

Mètode espectral en R L'objectiu principal del projecte és la realització d'una implementació en R del mètode espectral per a l'aprenentatge de màquines finites d'estats desenvolupat en els últims anys per, entre altres, investigadors del grup de recerca LARCA. Com a resultat d'aquesta implementació s'ha d'obtenir un paquet de rutines de R, o bé diversos paquets interrelacionats entre si.

Flux de treball El conjunt de funcions implementades ha de permetre satisfer un flux de treball bàsic, que dividim en tres fases ben diferenciades: preprocessament, processament (o nucli) i postprocessament o avaluació.

Eficiència El paquet que resulti del projecte ha de ser eficient. En particular, cal vèncer les dificultats que en aquest aspecte presenta el llenguatge R. Per això, s'explota al màxim la llibreria Rcpp, que permet integrar codi en C++ prèviament compilat en aplicacions de R.

Documentació S'ha d'obtenir un producte que pugui ser usat per persones d'arreu del món. Una gran fita seria la publicació dels paquets que en resultin al CRAN. Per això, és imprescindible redactar una bona documentació que segueixi els estàndards del llenguatge R. La documentació ha d'estar redactada, com a mínim, en anglès.

Aplicació Cal també dissenyar una aplicació del producte obtingut per mostrar el funcionament de les llibreries. Aquesta aplicació consisteix en una comparativa amb altres alternatives existents per a l'aprenentatge de màquines d'estats.

Memòria autocontinguda Per últim, es pretèn que la memòria del projecte sigui completa i autocontinguda, de manera que resulti comprensible i assequible per a qualsevol persona amb cultura científica i, en particular, per a qualsevol estudiant de matemàtiques o informàtica que tingui interés en la matèria estudiada.

1.3 Motivació

En aquest apartat expliquem tres exemples que poden servir de resposta a la pregunta: *Per a què aprendre màquines d'estats?*. I és que els autòmats finits són models sovint utilitzats per caracteritzar el comportament de diversos sistemes i processos, aportant l'avantatge d'ésser fàcilment interpretables.

Volem anar, però, una mica més enllà. Intentem donar una resposta a la pregunta, però sense entrar en gaires tecnicismes. Intentem que ens entengui qualsevol persona sense una formació estrictament científica que mostri un cert interés per saber què és el que estem fent en aquest projecte.

L'alemany Comencem amb un exemple senzill. Suposem, d'entrada, que no sabem alemany. Gens d'alemany. Res més enllà del fet que sembla que és des d'allà des d'on decideixen quant han de costar els nostres estudis, i si hi tenim dret o no, i el sou que haurem de cobrar quan siguem grans, i els serveis públics que ens mereixem, i... Bé, ja n'hi ha prou. Potser en sabem alguna cosa més: que *danke* vol dir *gràcies*; i comptar fins a tres, això sí: *ein*, *zwei*, *drei*.

Però si tenim al davant una paraula com *Zahlen*, sabríem dir si és alemanya o no?

D'entrada, no en sabríem dir res. Potser si ens donen alguna pista, com podria ser un text en alemany, d'unes 20000 paraules. Fàcil: Si *Zahlen* surt al text, ja hi som. Però si no... Alguna pista en podrem treure, del text. Per exemple: si hem vist moltes paraules acabades en *-en*, això dóna més probabilitats a què *Zahlen* sigui alemany. La proporció consonants-vocals de les paraules en alemany també ens pot donar alguna pista.

Probablement no tindríem prou informació com per donar un *sí* o un *no* rotund per a aquesta pregunta, però en tindríem prou com per assignar-hi un valor del 0 a l'1? Diguem-ne: la *probabilitat* que creiem que aquesta paraula té de ser alemanya.

Si ens preguntessin per la paraula *danke*, aquesta és fàcil, hi donaríem un 1. Si la paraula fos *pato*, probablement hi donéssim el 0. *Zahlen*... Potser n'acabaria sortint amb un 0.7 o així, depenent de què tracti el text.

D'on sortirien aquests valors? De processar el text, paraula a paraula, i fixar-nos en l'estructura d'aquestes: els prefixos, els sufixos, la freqüència de cada lletra, les partícules més recurrents, la longitud de cada paraula...

En aquest projecte treballarem sobre *models* que generaran uns *llenguatges* sobre uns *alfabets*. De la mateixa manera que aquí parlem de la *llengua alemanya* sobre l'*alfabet llatí* (amb alguna modificació), el nostre estudi tractarà sobre llenguatges *abstractes* sobre alfabets *abstractes* de n símbols.

Aquest exemple té cert interès per al projecte en el sentit que lliga l'aprenentatge de llenguatges abstractes generats per màquines d'estats i l'avaluació que fem *a posteriori* del model obtingut mitjançant la *perplexity*, en què assignem probabilitats a un conjunt de prova (vegeu Secció 3.7).

Navegant per la xarxa En aquest exemple introduirem els autòmats o màquines d'estats. Imaginem que som el *webmaster* d'una pàgina web i volem portar un registre de les accions que cada usuari du a terme a la nostra pàgina. Simplifiquem al màxim aquesta tasca, enregistrant només si, després de cada clic, l'usuari ha *entrat* a la nostra pàgina, s'hi ha *quedat* o si n'ha *sortit*.

A la Figura 1 mostrem un graf amb la possible interacció de l'usuari amb la pàgina. Els nodes són les pàgines dins del mateix web (en aquest cas, només són tres), i els diferents tipus d'arestes simbolitzen les diferents accions:

- L'aresta verda indica el punt d'entrada a la pàgina, la pàgina principal o *index*.
- Les arestes vermelles puntejades indiquen els punts de sortida de la pàgina (des de qualsevol pàgina es podria tancar el navegador).
- La resta d'arestes indiquen interaccions que fan que l'usuari es quedi dins del mateix web.

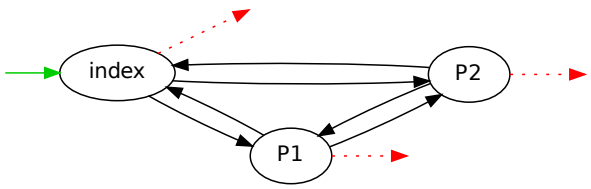


Figura 1: Exemple de navegació

L'esquema de la Figura 1 és, de fet, un autòmat no determinista sobre el llenguatge format pels símbols *entrar*, *quedar-se* i *sortir* (vegeu Secció 2.2).

Si escrivim e quan l'usuari entra a la pàgina, q quan s'hi queda i s quan en surt, qualsevol visita a la pàgina tindrà la mateixa estructura: una e seguida de qualsevol nombre de símbols q i acabada en una s. L'autòmat de la Figura 1 genera el llenguatge format per totes les *paraules* que comencen per e i acaben en s que tenen un nombre indeterminat de lletres q entre la e inicial i la s final.

En aquest cas particular, aprendre el llenguatge generat per aquest autòmat seria una tasca molt i molt fàcil.

Predint el comportament humà Suposem ara que la nostra pàgina web és una important pàgina de compra *on-line*, amb un gran volum d'usuaris i visites. Imaginem que podem predir quina és la intenció de cada usuari quan visita el nostre web: si busca un objecte en concret, si està buscant algun regal –qualsevol regal– per al dia de la mare o si, senzillament, està navegant per la xarxa i ha anat a parar a la nostra pàgina.

Amb aquesta informació en el nostre poder, el nostre servidor podria respondre de manera dinàmica a l'usuari: si està buscant un objecte en concret, li donaríem prioritat màxima enfront d'una possible sobrecàrrega del servidor, intentant no perdre una venda segura; si busca qualsevol regal, li faríem aparèixer ofertes diverses d'objectes relacionats; si és un navegant sense rumb, intentaríem convèncer-lo amb alguna oferta sucosa.

Modelant adientment la informació que ens proporciona cadascuna de les interaccions de l'usuari amb la nostra pàgina, podríem arribar a obtenir aquesta informació. L'usuari es comportarà de manera diferent segons quina sigui la seva intenció, i és aquí on entra l'aprenentatge de màquines d'estats.

El comportament humà davant d'una pàgina web es pot intentar modelitzar amb un autòmat. Cada clic seria un símbol de l'alfabet, i els estats finals serien aquells moments en què l'usuari marxa de la pàgina, havent comprat o no. Amb la informació adient, modelada de manera correcta i utilitzada de manera òptima, podríem ser capaços d'encaminar l'usuari cap aquells estats en què es maximitzi el nostre benefici. I per conèixer l'autòmat que hi ha darrere de les interaccions dels usuaris, faríem servir l'aprenentatge de màquines d'estats.

Altres exemples Altres exemples menys didàctics i més aplicats a la recerca són els següents casos reals en què s'utilitzen models basats en autòmats.

- Modelització de seqüències de proteïnes d'ADN en el camp de la bioinformàtica.
- Recerca de patrons sonors per al processat de la parla.
- Desenvolupament de regles morfològiques i fonològiques per al processament del llenguatge natural.
- En física, modelització de processos mecànics.
- Detecció d'anomalies per preveure intrusions en sistemes informàtics.
- Modelització de l'estructura dels diferents estils musicals per a la seva classificació.

1.4 Planificació i anàlisi econòmica del projecte

Aquest apartat engloba la planificació del projecte i la seva anàlisi econòmica.

1.4.1 Planificació del projecte

Aquest projecte s'ha desenvolupat en vuit mesos, entre octubre de 2012 i principis de juny de 2013, amb un ritme de treball mitjà de 20 hores setmanals entre octubre i març, i de 25 hores setmanals durant els mesos d'abril i maig.

El desenvolupament del projecte es pot dividir, bàsicament, en tres grans fases:

- Documentació, aprenentatge i anàlisi
- Desenvolupament i prova
- Redacció de la memòria

Si bé és natural dur a terme les tres fases del projecte seqüencialment, un cop una d'elles està prou avançada s'ha pogut començar, en paral·lel, la següent fase. A la Taula 1 mostrem la distribució horària *aproximada* de les tasques al llarg de tot el projecte.

Tot seguit, descrivim cadascuna de les fases del projecte.

Documentació, aprenentatge i anàlisi Aquesta primera fase considera la comprensió i l'enfocament del problema. D'una banda, la recerca de documentació, tant sobre el mètode espectral com sobre el llenguatge R: lectura d'articles, consulta de bibliografia i assistència a xerrades.

D'altra banda, incloem en aquesta primera part la subfase d'aprenentatge: Comprensió del mètode espectral, especialment mitjançant l'experimentació amb exemples senzills, i presa de contacte i pràctica amb el llenguatge R, així com elecció de les seves llibreries més adients.

Englobem també dins de la primera fase l'anàlisi del problema de cara a realitzar la implementació del mètode espectral de la millor manera possible dins d'un projecte d'enginyeria informàtica.

Per últim, incloem dins d'aquesta primera fase tots allò relacionat amb la gestió del projecte, com ara les reunions amb els tutors i la burocràcia (inscripció, matrícula, reserva de sales, etc).

Desenvolupament i prova Incloem en aquesta fase la implementació del mètode espectral en R i la prova de les llibreries resultants. D'aquesta part del projecte ha de sortir un conjunt de rutines en R que ha de ser complet, senzill i intuïtiu alhora.

També forma part d'aquesta segona fase l'experimentació amb les rutines obtingudes sobre un conjunt de models de prova.

Fase final: redacció de la memòria La redacció d'aquesta memòria és imprescindible per a l'avaluació del projecte, i ha de reflectir la totalitat del treball desenvolupat. Un dels objectius del projecte és que la memòria sigui autocontinguda.

Incloem en aquesta fase final del projecte la redacció de la documentació del paquet desenvolupat en el format estàndar de R i la preparació de la defensa a realitzar el dia 10 de juny.

1.4.2 Anàlisi econòmica del projecte

El món de la consultoria tecnològica, en què podriem encabir aquest projecte, es mou econòmicament a partir de la quantificació de les hores de dedicació que requerirà el projecte i l'avaluació del *preu* de cada hora de dedicació (incloent en la tarifa horària la mà d'obra i els recursos com ara l'electricitat o la tarifa d'internet). El preu del projecte prové de fer una senzilla multiplicació del nombre d'hores pel preu per hora.

Addicionalment, s'hi podrien afegir el preu de les llicències de software de tercers que s'hagin pogut requerir. En el cas d'aquest projecte, però, aquest concepte no aplica, ja que en tot moment s'ha fet servir software de distribució lliure i gratuïta.

El cost econòmic d'aquest projecte és, doncs, resultat de la multiplicació del nombre d'hores estimat de dedicació, 769 (vegeu Taula 1), multiplicat pel preu per hora, P.

Malauradament, en els temps actuals de crisi, retallades i preses de pèl constants per part d'aquells que acumulen el poder, el desenvolupador és l'última persona que hi tindrà alguna cosa a dir a l'hora de fixar quin ha de ser aquest preu P. L'empresari, afamat de becaris que facin feina bruta a un preu irrisori, farà tot el possible perquè $P \to 0$, sense que una anàlisi econòmica dels costos reals pugui influir gaire en el valor d'aquesta P.

Fixem, doncs, el preu d'aquest projecte en $769 \cdot P$, i deixem la determinació d'aquesta P com a exercici per al lector. La tasca no és trivial i requereix aptituds negociadores excepcionals.

Tasca	Total	Oct	Nov	Des	Gen	Feb	Mar	Abr	Mai	Jun
Fase inicial: Documentació, aprenentatge i anàlisi	**274**	**85**	**85**	**31**	**26**	**21**	**16**	**5**	**5**	**0**
Lectura i comprensió de documentació	120	55	30	10	10	10	5			
Anàlisi i experimentació prèvia amb el mètode espectral	25	10	15							
Aprenentatge del llenguatge R i llibreria Rcpp	70	10	25	15	10	5	5			
Assistència a xerrades i conferències	27	5	10	3	3	3	3			
Gestió del projecte	32	5	5	3	3	3	3	5	5	
Fase central: Desenvolupament i prova	**300**	**0**	**10**	**50**	**45**	**40**	**50**	**65**	**40**	**0**
Implementació del mètode espectral en R	150		5	30	25	20	25	35	10	
Proves unitàries de la correctesa de la implementació	90		5	20	20	20	15	10		
Proves integrades: Experimentació	60						10	20	30	
Fase final	**195**	**0**	**0**	**10**	**15**	**25**	**25**	**35**	**70**	**15**
Redacció de la memòria	165			10	15	25	25	35	50	5
Documentació del paquet en R	20								20	
Preparació de la defensa	10									10
Total	**769**	**85**	**95**	**91**	**86**	**86**	**91**	**105**	**115**	**15**

Taula 1: Planificació horària aproximada de les tasques del projecte

2 Coneixements previs

En aquesta secció es pretèn posar en context els coneixements previs necessaris per comprendre la teoria del mètode espectral. S'hi descriuen conceptes bàsics de teoria de llenguatges i autòmats, començant pel concepte més senzill d'*alfabet* o *símbol* i fins a arribar a descriure els objectes sobre els quals es desenvolupa l'aprenentatge del mètode espectral.

2.1 Conceptes bàsics

Alfabet Un *alfabet* Σ és un conjunt de símbols. Són alfabets els conjunts $A = \{a, b\}$, $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ i $W = \{\text{entrar}, \text{quedar-se}, \text{sortir}\}$, per exemple.

Símbol Anomenem *símbol* qualsevol element d'un alfabet.

Els conceptes d'*alfabet* i *símbol* són la base de la teoria de llenguatges així com els conceptes de *conjunt* i *element* són la base de les matemàtiques. No n'existeix una definició formal que no apel·li a la intuïció.

Paraula Una *paraula* és una successió finita i ordenada de símbols, $w = x_1 x_2 \cdots x_n$.

Clausura de Kleene La *clausura de Kleene* Σ^* és el conjunt format per totes les paraules que es poden formar amb els símbols d'un alfabet. Inclou la paraula buida, que se sol representar amb el símbol λ. Σ^* és un conjunt infinit numerable sempre i quan Σ no sigui buit.

Llenguatge Un llenguatge L és un subconjunt de Σ^*.

El concepte de llenguatge en si és massa ampli, per la qual cosa sovint ens referim a categories de llenguatges, com ara els llenguatges reconeguts per autòmats o els llenguatges reconeguts per gramàtiques.

2.2 Autòmats

2.2.1 Autómats finits deterministes i no-deterministes

Formalment, un *autòmat finit determinista* (DFA) és una tupla $\langle Q, \Sigma, \delta, q_0, F \rangle$, on:

- Q és un conjunt finit d'estats.
- Σ és un alfabet.
- $\delta : Q \times \Sigma \longrightarrow Q$ és la funció de transició.
- q_0 és l'estat inicial.
- $F \subset Q$ és el conjunt d'*estats acceptadors*.

Els autòmats se solen representar gràficament. A mode d'exemple, a la Figura 2 es mostra l'autòmat $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, amb:

- $Q = \{S1, S2\}$
- $\Sigma = \{a, b\}$
- $q_0 = S1$
- $F = \{S2\}$

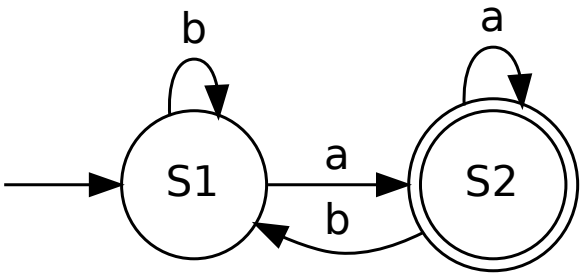


Figura 2: Autòmat finit determinista (DFA)

I la funció de transició δ, definida per:

$$\begin{array}{rccc} \delta: & Q \times \Sigma & \longrightarrow & Q \\ & (S1, a) & \longmapsto & S2 \\ & (S1, b) & \longmapsto & S1 \\ & (S2, a) & \longmapsto & S2 \\ & (S2, b) & \longmapsto & S1 \end{array}$$

Un autòmat *llegeix* una paraula $w = x_1 x_2 \cdots x_n$, generant una seqüència d'estats, $q_0, q_1, \ldots, q_n$. Si l'estat final pertany als estats acceptadors, $q_n \in F$, la paraula és *acceptada* per l'autòmat. Altrament, la paraula es *refusa*. Es diu informalment que un autòmat és un objecte matemàtic que pren com a entrada una paraula i decideix si acceptar-la o refusar-la.

En l'exemple de la Figura 2, la paraula $w = aabaa$ genera la seqüència d'estats $S1, S2, S2, S1, S2, S2$, per la qual cosa A *accepta* w. La paraula buida λ, en canvi, genera la seqüència d'un únic element: $S1$. Com que $S1 \notin F$, A *refusa* la paraula buida.

Llenguatge generat per un autòmat El *llenguatge generat per un autòmat* és el llenguatge format per totes aquelles paraules que són acceptades per l'autòmat.

El llenguatge acceptat per l'autòmat de la Figura 2 és el format per totes aquelles paraules del llenguatge $\{a, b\}$ que acaben en a, $\{w = x_1 x_2 \cdots x_n \in \{a, b\}^* | x_n = a\}$

L'autòmat finit determinista és el tipus d'autòmat més bàsic. A partir d'aquí, petites variacions en la definició produeixen una gran diversitat de tipus d'autòmats.

Per exemple, *els autòmats finits no deterministes* (NFA) són autòmats en què l'autòmat, en llegir un símbol, pot decidir anar a més d'un estat. A més a més, pot començar l'execució en *qualsevol* estat dins d'un *conjunt d'estats inicials*.

Per als NFAs, se substitueix la definició d'autòmat per $\langle Q, \Sigma, \sim, I, F \rangle$, expressió en què s'ha substituït l'estat inicial q_0 per un *conjunt d'estats inicials* I, i la funció de transició δ per una *relació de transició* $\sim$.

Un NFA accepta una paraula si, i només si, existeix una seqüència d'estats que acaba en un estat acceptador.

L'autòmat de la Figura 3, per exemple, és no-determinista perquè des de l'estat $S2$ pot decidir si anar a l'estat $S1$ o quedar-se a $S2$ si el símbol llegit és una b. Concretament, el llenguatge acceptat per aquest autòmat és el format per totes les paraules de $\{a, b\}^*$ que contenen almenys una a,

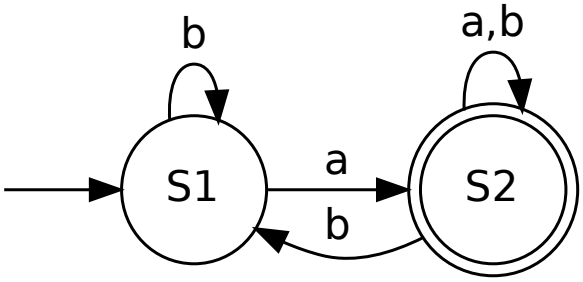


Figura 3: Autòmat finit no determinista (NFA)

$\{w = x_1x_2\cdots x_n \in \{a,b\}^* |\quad \exists i \in \{1\ldots n\}, x_i = a\}$.

2.2.2 Automàts finits probabilistes

Un *autòmat finit probabilista* (PFA), és un autòmat no determinista en què cada transició i cada estat inicial tenen assignada una probabilitat.

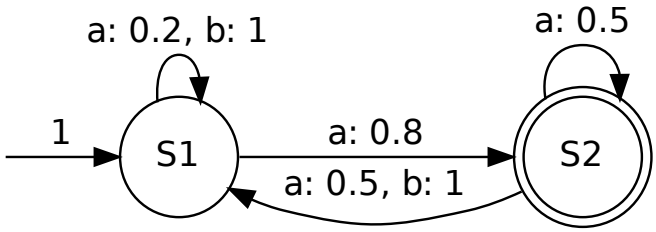


Figura 4: Autòmat finit probabilista (PFA)

A la Figura 4 mostrem un exemple de PFA. L'estat inicial serà $S1$ amb probabilitat 1. A partir d'aquí, si es llegeix una b el següent estat serà $S1$, però si es llegeix una a el següent estat serà $S2$ amb una probabilitat de 0.8 i $S1$ amb una probabilitat de 0.2. Per acceptar la paraula a cal, doncs, que la seqüència d'estats sigui: $S1, S2$ i la probabilitat d'aquest esdeveniment és de 0.8.

Suposem que llegim la paraula *abaa*. Les possibles seqüències acceptadores de la paraula són:

- $S1, S1, S1, S1, S2$, amb probabilitat $1 \cdot 0.2 \cdot 1 \cdot 0.2 \cdot 0.8 = 0.032$
- $S1, S1, S1, S2, S2$, amb probabilitat $1 \cdot 0.2 \cdot 1 \cdot 0.8 \cdot 0.5 = 0.08$
- $S1, S2, S1, S1, S2$, amb probabilitat $1 \cdot 0.8 \cdot 1 \cdot 0.2 \cdot 0.8 = 0.128$
- $S1, S2, S1, S2, S2$, amb probabilitat $1 \cdot 0.8 \cdot 1 \cdot 0.8 \cdot 0.5 = 0.32$

Per la qual cosa, la paraula *abaa* té una probabilitat de $0.032 + 0.08 + 0.128 + 0.32 = 0.56$ de ser acceptada per l'autòmat. D'altra banda, per exemple, la paraula *aab* serà sempre refusada per l'autòmat, perquè no existeix cap seqüència acceptadora.

Podem expressar les probabilitats inicials vectorialment i les probabilitats de transició matricialment. Per a l'exemple de la Figura 4:

$$\alpha_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, A_a = \begin{pmatrix} 0.2 & 0.8 \\ 0.5 & 0.5 \end{pmatrix}, A_b = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

Observem que un autòmat probabilista *indueix* una funció sobre el conjunt de totes les paraules de l'alfabet Σ^*, $f_{\mathcal{A}} : \Sigma^* \longrightarrow \mathbb{R}$. En l'exemple de la Figura 4, $f_{\mathcal{A}}(a) = 0.8$, $f_{\mathcal{A}}(abaa) = 0.56$ i $f_{\mathcal{A}}(aab) = 0$.

Generalment, però, amb els autòmats probabilistes no es parla d'*acceptar* paraules, sinó de *generar-les*. Modifiquem lleugerament l'autòmat de la Figura 4 per obtenir l'autòmat de la Figura 5.

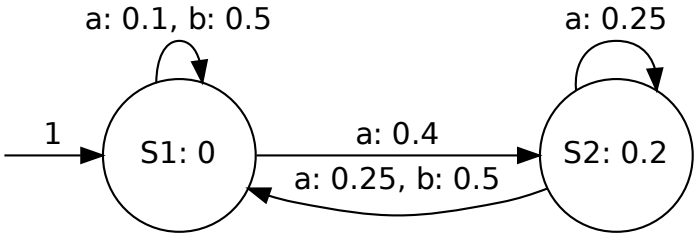


Figura 5: Autòmat finit probabilista (PFA)

Noteu que a l'autòmat de la Figura 5, a més de modificar els valors de les probabilitats de les transicions, hi hem afegit uns valors que acompanyen les etiquetes dels nodes. Aquests valors són les *probabilitats finals*, o probabilitats d'acabar la paraula en aquell estat. En l'exemple de la figura, l'estat $S1$ no és generador en cap cas, i l'estat $S2$ donarà per acabada la paraula amb probabilitat 0.2.

Els valors que acompanyen les transicions prenen un altre significat: en l'exemple de la Figura 5, des de l'estat $S1$ hi ha una probabilitat de 0.1 de generar una a i quedar-se a l'estat $S1$, una probabilitat de 0.4 de generar una a i passar a l'estat $S2$ i una probabilitat de 0.5 de generar una b i quedar-se a l'estat $S1$.

Amb aquesta concepció diferent dels PFAs, la probabilitat de les paraules de l'exemple anterior varia. La probabilitat de generar la paraula a, per exemple, és el producte de la probabilitat de generar una a en l'únic estat inicial, $S1$, passant a l'únic estat generador, 0.4 per la probabilitat que l'estat $S2$ generi la paraula, 0.2. Tenim, doncs, $f_{\mathcal{A}}(a) = 0.08$.

Noteu que, de la mateixa manera que representem les probabilitats inicials mitjançant un vector α_0, podem representar les probabilitats finals mitjançant un vector que denotarem α_∞.

Formalment, doncs, un PFA es pot veure com una tupla $\mathcal{A} = \langle \alpha_1, \alpha_\infty, \{A_\sigma\}_{\sigma \in \Sigma} \rangle$, on:

- $\alpha_0 = (\alpha_0^1, \ldots, \alpha_0^n) \in \mathbb{R}^n$ és el vector amb les probabilitats inicials. Així, α_0^i és la probabilitat que l'estat inicial sigui l'estat i.
- $\alpha_\infty = (\alpha_\infty^1, \ldots, \alpha_\infty^n) \in \mathbb{R}^n$ és el vector amb les probabilitats finals. Així, α_∞^i és la probabilitat que l'estat i accepti la paraula.
- $A_\sigma = (a_{ij}) \in \mathbb{R}_{n \times n}$ és la matriu que conté les probabilitats de transició entre estats. Així, a_{ij} és la probabilitat de transició de l'estat i a l'estat j.

Pel seu caràcter probabilista, un PFA $\mathcal{A} = \langle \alpha_1, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \rangle$ satisfà les següents propietats:

- $\alpha_0^i, \alpha_\infty^i, a_{ij} \geq 0 \quad \forall i, j \in \{1, \ldots, n\}$
- $\sum_{i=1}^{n} \alpha_0^i = 1$
- $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i \in \{1, \ldots, n\}$

Observem que, amb aquesta notació, podem expressar la funció induïda per $\mathcal{A}$ de la manera següent:

$$f_\mathcal{A}(x_1 \cdots x_m) = \alpha_0^\top A_{x_1} \cdots A_{x_m} \alpha_\infty$$

La funció induïda per un PFA (entés com a generador) és, de fet, una funció de probabilitat definida sobre el conjunt Σ^*, ja que satisfà $\sum_{w\in\Sigma^*} f_\mathcal{A} = 1$.

A partir de la funció de probabilitat d'una paraula $f_\mathcal{A}(w) = \mathbf{Pr}_\mathcal{A}(w)$ podem definir també la probabilitat d'un prefix u, $\mathbf{Pr}_\mathcal{A}(u\Sigma^*)$. Si $\mathcal{A} = \langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \rangle$, aleshores

$$\mathbf{Pr}_\mathcal{A}(u\Sigma^*) = \mathbf{Pr}_\mathcal{A}(x_1 \cdots x_m \Sigma^*) = \sum_{i=1}^{|\Sigma|} \left(\alpha_0^\top A_{x_1} \cdots A_{x_m}\right)^i$$

Aquesta última concepció de PFA que hem descrit és la que utilitzarem en la resta del projecte.

Autòmat finit probabilista determinista (DPFA) Els DPFA són un cas particular de PFA, però en versió determinista. És a dir, de cada estat surt una única transició per a cada símbol (que pot tenir probabilitat 0).

2.2.3 *Weighted Automata*

Un *autòmat finit amb pesos*, en anglès *weighted automaton* (WA), es pot entendre com una generalització del concepte de PFA. Un WA és, de fet, un PFA al qual no s'hi exigeixen les restriccions imposades als PFAs pel seu caràcter probabilista.

Formalment, un WA és una tupla $\mathcal{A} = \langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \rangle$, amb:

- $\alpha_0 = (\alpha_0^1, \ldots, \alpha_0^n) \in \mathbb{R}^n$

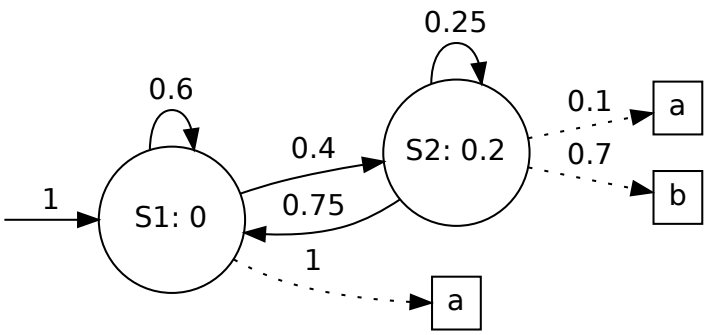


Figura 6: *Hidden Markov model* (HMM)

- $\alpha_\infty = (\alpha_\infty^1, \ldots, \alpha_\infty^n) \in \mathbb{R}^n$
- $A_\sigma = (a_{ij}) \in \mathbb{R}_{n \times n}$

Anàlogament a la funció induïda per un PFA, la funció induïda per un WA $\mathcal{A}$ és:

$$\begin{array}{rccl} f_{\mathcal{A}}: & \Sigma^* & \longrightarrow & \mathbb{R} \\ & x_1 \cdots x_m & \longrightarrow & f_{\mathcal{A}}(x_1 \cdots x_m) = \alpha_0^\top A_{x_1} \cdots A_{x_m} \alpha_\infty \end{array}$$

2.2.4 *Hidden Markov Models*

Un *model ocult de Markov*, en anglès *hidden Markov Model* (HMM), es pot entendre com un PFA en què els símbols, en comptes d'estar assignats a les transicions i ser emesos (generats) en efectuar-se un canvi d'estat, s'emeten des dels mateixos estats.

A la Figura 6 tenim un exemple de HMM. Des de l'estat $S1$ s'emet sempre una a i hi ha una probabilitat de 0.4 de passar a l'estat $S2$. A l'estat $S2$ hi ha una probabilitat de 0.2 d'acabar la paraula, una probabilitat de 0.1 d'emetre una a i una probabilitat de 0.7 d'emetre una b.

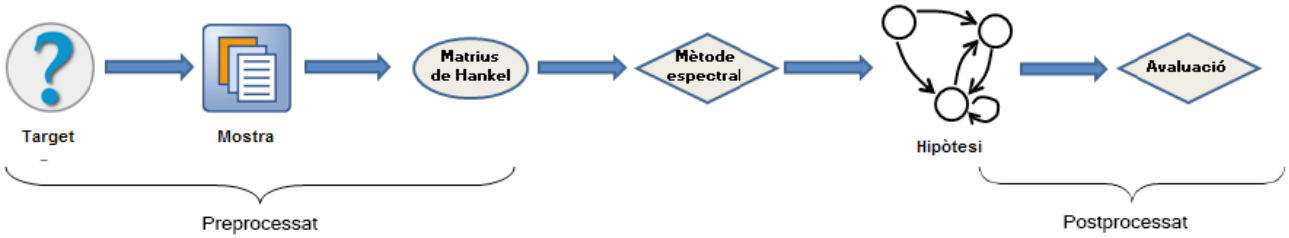
3 Fonaments teòrics del mètode espectral

L'objectiu del mètode espectral és l'aprenentatge de màquines d'estats probabilistes a partir d'una mostra. Es parteix d'un model, generalment desconegut, que anomenarem *objectiu* o *target* i s'acaba obtenint, via el mètode espectral, un model *hipòtesi*, la qualitat del qual cal avaluar.

A grans trets, el mètode espectral consisteix en el següent:

1. Obtenció d'una mostra
2. Pretractament de la mostra
3. Obtenció d'una base
4. Càlcul de les matrius de Hankel en la base escollida
5. Descomposició en valors singulars de la matriu de Hankel. Determinació del nombre d'estats
6. Construcció de l'autòmat hipòtesi
7. Avaluació de l'autòmat hipòtesi

Podem representar el flux de treball del mètode espectral gràficament com a la figura següent. Podem subdividir-lo en tres fases: preprocessament, mètode espectral i avaluació o postprocessament.



Durant aquest capítol explicarem els fonaments teòrics del mètode espectral. En el següent capítol, desenvoluparem detalladament l'algorisme tipus per obtenir un model a partir d'unes dades.

3.1 Funcions definides sobre strings

En el Capítol 2 ja hem comentat que els autòmats probabilistes, en particular, i els *weighted automata*, en general, indueixen funcions que estan definides sobre el domini Σ^*.

En aquest projecte descrivim el mètode espectral per a l'aprenentatge de màquines d'estats. Concretament, a partir d'una mostra de la funció computada per un autòmat, *aprenem* quina és aquesta funció i posteriorment trobem un autòmat que la indueix o computa.

Formalment, a partir de la informació que ens proporciona una mostra $\mathcal{M} \subset \Sigma^*$, aproximarem una funció $f : \Sigma^* \longrightarrow \mathbb{R}$ i construirem un autòmat $\mathcal{A} = \langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \rangle$ tal que $f = f_\mathcal{A}$.

Ara bé, l'autòmat $\mathcal{A}$ que obtinguem no té per què estar encarat a aproximar la funció que computa l'autòmat $\mathcal{B} = \langle \beta_0, \beta_\infty, \{B_\sigma\}_{\sigma\in\Sigma} \rangle$ original. Generalment ens trobarem en el cas en què $\mathcal{B}$ és un autòmat probabilista que computa la funció $f_\mathcal{B}(w) = \mathbf{Pr}_\mathcal{B} w$, però el nostre mètode d'aprenentatge no té per què limitar-se a aproximar $f_\mathcal{B}$.

En aquest sentit, ens referirem tot sovint a *aprendre sobre strings*, *aprendre sobre prefixos* o *aprendre sobre substrings*. Tot dependrà de quina informació extraguem de la mostra.

Strings Aprendrem sobre *strings* si calculem la probabilitat de trobar una determinada paraula w dins de la mostra $\mathcal{M}$, $\mathbf{Pr}_\mathcal{M}(w)$.

Prefixos Aprendrem sobre prefixos si calculem la probabilitat de trobar una determinada paraula u com a prefix d'una paraula de la mostra, $\mathbf{Pr}_\mathcal{M}(u\Sigma^*)$.

Substrings Aprendrem sobre *substrings* si calculem l'esperança del nombre de vegades que trobarem la partícula x com a *substring* d'una paraula w de la mostra, $\mathbb{E}\left(|w|_x\right)$.

3.2 La matriu de Hankel. Bases

Sigui Σ un alfabet finit amb m símbols. Sigui $\lambda \in \Sigma^*$ la paraula buida. Sigui $f : \Sigma^* \longrightarrow \mathbb{R}$ una funció definida sobre strings.

3.2.1 Matriu de Hankel

La *matriu de Hankel* de f és $H_f : \Sigma^* \times \Sigma^* \to \mathbb{R}$, definida per $H_f(u, v) = f(uv)$. Es pot veure com una matriu infinita indexada per mots $u, v \in \Sigma^*$. Anomenem *prefixos* els u i *sufixos* els v.

Per exemple, si $\Sigma = \{a, b\}$, la matriu de Hankel d'una funció $f : \{a, b\}^* \to \mathbb{R}$ és

$$\begin{array}{c} \\ \lambda \\ a \\ b \\ aa \\ ab \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} \lambda & a & b & aa & ab & \cdots \end{array} \\ \begin{pmatrix} f(\lambda) & f(a) & f(b) & f(aa) & f(ab) & \cdots \\ f(a) & f(aa) & f(ab) & f(aaa) & f(aab) & \cdots \\ f(b) & f(ba) & f(bb) & f(baa) & f(bab) & \cdots \\ f(aa) & f(aaa) & f(aab) & f(aaaa) & f(aaab) & \cdots \\ f(ab) & f(aba) & f(abb) & f(abaa) & f(abab) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}$$

Observem que la matriu de Hankel emmagatzema informació redundant. Seguint amb l'exemple anterior, tenim que el valor de $f(w)$ per $w = ab$ apareix 3 vegades a H_f:

$$\begin{array}{c} \\ \lambda \\ a \\ b \\ aa \\ ab \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} \lambda & a & b & aa & ab & \cdots \end{array} \\ \begin{pmatrix} & & & & f(ab) & \\ & & f(ab) & & & \\ & & & & & \\ & & & & & \\ f(ab) & & & & & \\ & & & & & \end{pmatrix} \end{array}$$

Més en general, el valor de $f(w)$, per $w \in \Sigma^*$, apareix $|w| + 1$ vegades a la matriu de Hankel. Si $w = x_1 \cdots x_m$:

$$\begin{array}{c} \\ \lambda \\ x_1 \\ x_1x_2 \\ \vdots \\ x_1 \cdots x_{m-1} \\ x_1 \cdots x_m \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} \lambda & x_m & \cdots & x_3 \cdots x_m & x_2 \cdots x_m & x_1 \cdots x_m & \cdots \end{array} \\ \begin{pmatrix} & & & & & f(w) & \\ & & & & f(w) & & \\ & & & f(w) & & & \\ & & \cdots & & & & \\ & f(w) & & & & & \\ f(w) & & & & & & \\ & & & & & & \end{pmatrix} \end{array}$$

Rang de f Sigui $f : \Sigma^* \longrightarrow \mathbb{R}$. El *rang* de f és el rang de la matriu H_f. Noteu que $rang(f)$ podria valer infinit.

Sub-blocs de la matriu de Hankel Podem considerar sub-blocs de la matriu de Hankel. Així, si $\mathcal{U}, \mathcal{V} \subset \Sigma^*$ són subconjunts de prefixos i sufixos, la matriu H, definida per

$$\begin{array}{rccl} H: & \mathcal{U} \times \mathcal{V} & \longrightarrow & \mathbb{R} \\ & (u, v) & \longrightarrow & H(u, v) = f(uv) \end{array}$$

és un sub-bloc de la matriu de Hankel $H_f : \Sigma^* \times \Sigma^* \to \mathbb{R}$.

Un sub-bloc H d'una matriu de Hankel H_f satisfà les següents propietats:

- $rang(H) \leq rang(H_f)$
- Si $|\mathcal{U}| = p$ i $|\mathcal{V}| = s$, aleshores $H \in \mathbb{R}^{p \times s}$.

3.2.2 Bases. Les matrius H_σ

Un parell de conjunts de prefixos i sufixos $(\mathcal{P}, \mathcal{S})$ és una *base* si es té $rang(H) = rang(H_f)$.

Sigui $\sigma \in \Sigma$. Es defineix $H_\sigma \in \mathbb{R}^{p \times s}$ com la matriu que a cada parell prefix-sufix (u, v) li assigna el valor $H_\sigma(u, v) = f(u\sigma v)$.

Fixada una base $(\mathcal{P}, \mathcal{S})$, sovint ens referirem a les matrius $H, \{H_\sigma\}_{\sigma \in \Sigma}$ com les *matrius de Hankel del model en la base* $(\mathcal{P}, \mathcal{S})$, tenint en compte que les matrius $\{H_\sigma\}$ són sub-blocs particulars de la matriu de Hankel.

3.2.3 Rang de la funció calculada per una màquina d'estats finita

Sigui $f : \Sigma^* \longrightarrow \mathbb{R}$. A la Secció 3.2.1 hem definit $rang(f) = rang(H_f)$, on H_f és la matriu de Hankel de f. Aleshores,

$$rang(f) = r \Longleftrightarrow f = f_{\mathcal{A}} \text{ per a cert WA } \mathcal{A} = \left\langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma \in \Sigma} \right\rangle \text{ amb } r \text{ estats}$$

Aquest resultat ens garanteix que podem trobar una base finita que generi un sub-bloc de la matriu de Hankel de rang màxim.

3.2.4 Matriu de Hankel d'una mostra

En els apartats anteriors ens hem referit a la *matriu de Hankel* com una matriu inherent a la funció i al model que la computa. Ara bé, model i funció computada seran desconeguts, per la qual cosa la matriu de Hankel real també serà desconeguda. Tota la informació del model de què disposarem haurem d'extreure-la a partir d'una mostra $\mathcal{M} \subset \Sigma^*$.

Considerem un autòmat $\mathcal{A}$ que computa una funció $f_{\mathcal{A}}$ de probabilitat sobre *strings*, $f_{\mathcal{A}}(w) = \mathbf{Pr}_{\mathcal{A}}(w)$. Si bé la funció de probabilitat sobre tot Σ^* serà desconeguda, el que sí podrem calcular és la funció de probabilitat sobre la mostra, $\mathbf{Pr}_{\mathcal{M}}(w)$.

Ens referirem a la *matriu de Hankel d'una mostra* per referir-nos a la matriu de Hankel de l'aproximació a una funció a través de la informació proporcionada per la mostra.

Per diferenciar la matriu de Hankel obtinguda a partir d'una mostra de la matriu de Hankel real, escriurem H_f per a la matriu de Hankel d'una mostra i $\widetilde{H_f}$ per a la matriu de Hankel real. Sovint utilitzarem simplement el terme *matriu de Hankel* referint-nos a la matriu de Hankel de la mostra.

3.3 Descomposició en valors singulars

L'eina fonamental del mètode espectral, la qual dóna nom al mètode, és la descomposició en valors singulars, més coneguda per les seves sigles o el seu nom en anglès: *SVD*, *Singular Value Decomposition*.

La seva importància en el mètode prové del fet que ens permetrà descomposar la matriu de Hankel, H, en el producte de dues matrius rectangulars i una matriu diagonal. Així, podrem expressar

$$H = U\Lambda V^\top$$

on

- $U \in R_{m\times k}$ i $V^\top \in R_{k\times n}$
- $\Lambda \in R^+_{k\times k}$ matriu diagonal (quadrada) amb valors estrictament positius.
- $rang(H) = k$

3.3.1 Teorema de descomposició en valors singulars

Sigui $M \in R_{m\times n}$ una matriu amb valors reals. Aleshores, existeix una factorització de la forma

$$M = U\Lambda V^\top$$

amb

- $U \in R_{m\times m}$ i $V \in R_{n\times n}$ matrius ortogonals.
- $\Lambda \in R^+_{m\times n}$ matriu diagonal (rectangular) amb valors no negatius.

Anomenem *descomposició en valors singulars*, o *SVD*, la factorització $M = U\Lambda V$.

Addicionalment, és freqüent assumir que els valors de la diagonal de Λ estan ordenats decreixentment. Així, si (per $m \leq n$), escrivim

$$\Lambda = \left(\begin{array}{cccc|ccc} \lambda_1 & & & & 0 & \cdots & 0 \\ & \lambda_2 & & & \vdots & & \vdots \\ & & \ddots & & \vdots & & \vdots \\ & & & \lambda_m & 0 & \cdots & 0 \end{array}\right)$$

i assumirem $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$.

Anomenem *valors singulars* els valors λ_i de la diagonal de Λ.

3.3.2 Notació alternativa

Sigui $M \in R_{m\times n}$ una matriu amb valors reals de rang k.

Sovint s'utilitza una notació alternativa per expressar la SVD. Podem expressar la factorització $M = \hat{U}\hat{\Lambda}\hat{V}^\top$, on:

- $\hat{U} \in R_{m\times k}$ i $\hat{V}^\top \in R_{k\times n}$
- Els k vectors columna $u_1, \ldots, u_k$ de $\hat{U}$ són unitaris i ortogonals dos a dos.
- Els k vectors columna $v_1, \ldots, v_k$ de $\hat{V}$ són unitaris i ortogonals dos a dos.
- $\hat{\Lambda} \in R^+_{k\times k}$ matriu diagonal (quadrada) amb valors estrictament positius.

Obtenir aquesta representació de la SVD és senzill si tenim en compte que

$$rang(M) = k \Leftrightarrow rang(\Lambda) = k \Leftrightarrow \lambda_i = 0 \quad \forall i > k$$

i expressem la factorització $M = U\Lambda V$ anomenant u_j, v_j els vectors columna de U, V. Aleshores,

$$M = \left(\begin{array}{cccccc} \uparrow & & \uparrow & \uparrow & & \uparrow \\ u_1 & \cdots & u_k & u_{k+1} & \cdots & u_m \\ \downarrow & & \downarrow & \downarrow & & \downarrow \end{array}\right) \left(\begin{array}{cccc|ccc} \lambda_1 & & & & 0 & \cdots & 0 \\ & \ddots & & & \vdots & & \vdots \\ & & \lambda_k & & & & \\ & & & 0 & & & \\ & & & & \vdots & & \vdots \\ & & & & 0 & \cdots & 0 \end{array}\right) \left(\begin{array}{ccc} \leftarrow & v_1 & \rightarrow \\ & \vdots & \\ \leftarrow & v_k & \rightarrow \\ \leftarrow & v_{k+1} & \rightarrow \\ & \vdots & \\ \leftarrow & v_n & \rightarrow \end{array}\right)$$

i, simplificant l'expressió:

$$M = \left(\begin{array}{ccc} \uparrow & & \uparrow \\ u_1 & \cdots & u_k \\ \downarrow & & \downarrow \end{array}\right) \left(\begin{array}{ccc} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{array}\right) \left(\begin{array}{ccc} \leftarrow & v_1 & \rightarrow \\ & \vdots & \\ \leftarrow & v_k & \rightarrow \end{array}\right) = \hat{U}\hat{\Lambda}\hat{V}$$

amb $\hat{U}$, $\hat{V}$ i $\hat{\Lambda}$ satisfent les propietats abans esmentades.

3.4 Pseudoinversa

El mètode espectral requereix en certs moments la inversió de matrius que no tenen per què ser invertibles. Per a certs casos, de fet, ni tan sols seran quadrades. Aquesta aparent contradicció es resol per al mètode espectral amb el que es coneix com a *matriu pseudoinversa*, una matriu que garanteix *algunes* de les propietats de la inversa d'una matriu invertible, entre les quals es troben les que requereix el mètode espectral.

La *pseudoinversa* o *inversa generalitzada* d'una matriu qualsevol A és una generalització de la inversió per a matrius no necessàriament quadrades. Si bé hi ha diferents descripcions de la pseudoinversa d'una matriu, la més usada és la *pseudoinversa de Moore-Penrose*, que descrivim tot seguit.

De fet, sovint direm només *pseudoinversa* referint-nos a la pseudoinversa de Moore-Penrose.

3.4.1 Pseudoinversa de Moore-Penrose

Sigui $A \in \mathbb{R}_{m\times n}$. La *pseudoinversa de Moore-Penrose* de A és una matriu $A^+ \in \mathbb{R}_{n\times m}$ que satisfà les següents quatre propietats:

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. AA^+ és simètrica: $(AA^+)^\top = AA^+$
4. A^+A és simètrica: $(A^+A)^\top = A^+A$

Propietats bàsiques

- La pseudoinversa existeix i és única.
- Si A és invertible, aleshores $A^{-1} = A^+$
- $(A^+)^+ = A$
- $(A^+)^\top = (A^\top)^+$
- Si les columnes de A són linealment independents, aleshores $A^+A = I_n$

- Si les files de A són linealment independents, aleshores $AA^+ = I_m$
- $(\alpha A)^+ = \frac{1}{\alpha} A^+$

3.5 Construcció de l'autòmat hipòtesi a partir de la matriu de Hankel

L'objectiu és obtenir un autòmat $\mathcal{A} = \langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma \in \Sigma} \rangle$ a partir de la informació emmagatzemada a la matriu de Hankel.

Siguin $H, \{H_\sigma, \sigma \in \Sigma\}$ les matrius de Hankel, $H = U \Lambda V^\top$ la descomposició en valors singulars de H i V_n la matriu formada per les n primeres columnes de V. Construïm la màquina hipòtesi $\mathcal{A}$ de la següent manera:

$$\mathcal{A} = \langle H[\lambda, \cdot] \cdot V_n, (H \cdot V_n)^+ \cdot H[\cdot, \lambda], \left\{ (H \cdot V_n)^+ \cdot H_\sigma \cdot H[\cdot, \lambda] \right\}_\sigma \in \Sigma \rangle$$

Aquest autòmat és l'autòmat de n estats que minimitza l'error respecte de l'autòmat real.

3.6 Avaluació per *word error rate*

El *word error rate* (WER) és una mètrica utilitzada sovint per a l'avaluació de sistemes relacionats amb el processament del llenguatge natural. Es basa en calcular la taxa d'error en predir quina és la paraula a utilitzar donat un context determinat.

Per avaluar els models apresos mitjançant el mètode espectral proposem una versió adaptada del *word error rate*. Al cap i a la fi, podem considerar el WER com una mètrica que necessita només tres ingredients per a la seva definició: una màquina, una mostra i una regla de predicció.

La regla de predicció consisteix en determinar què és el que s'ha de predir (en la concepció original del WER podem voler predir una paraula, una categoria gramatical o una estructura sintàctica, per exemple) i en quin context (havent llegit la resta de l'oració, la resta del text, només les paraules prèvies, etc).

En el nostre cas, es farà la predicció símbol a símbol. Amb la predicció de cada símbol dins de d'una mateixa paraula calcularem el WER de la paraula en qüestió, que també podríem anomenar, de fet, *symbol error rate*. La mitjana dels WERs de cadascuna de les paraules de la nostra determinarà el *word error rate* de la màquina apresa. Aquesta és la nostra regla de predicció per al WER.

Formalment, sigui w una paraula, $w = x_1 x_2 \dots x_n$. Com que la predicció es farà símbol a símbol, denotarem per σ_k el símbol predit a la posició k. Es proposen les següents dues regles de predicció:

- $\sigma_k = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\mathbf{Pr} \left(x_1 \cdots x_{k-1} \sigma \Sigma^* \right) \right)$
- $\sigma_k = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\mathbf{Pr} \left(x_1 \cdots x_{k-1} \sigma x_{k+1} x_n \right) \right)$

Observem que, en la primera de les opcions, prenem aquell símbol σ_k que tingui probabilitat màxima d'aparèixer després del prefix llegit fins a la posició k-èsima, independentment de què vingui després d'aquest símbol. Amb la segona regla de predicció, escollim aquell símbol σ_k que tingui millors opcions per a encaixar en la resta de la paraula, tenint en compte cada tota la resta de la paraula excepte el símbol k-èsim.

Dit d'una altra manera, el *context* per a la primera regla de predicció són els símbols previs al símbol a predir dins de la mateixa paraula, mentre que en la segona regla de predicció el context és tota la resta de la paraula.

El WER de la paraula w és, sigui quina sigui la regla de predicció per a cada símbol:

$$WER_w = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{x_i \neq \sigma_i\}}$$

El WER sobre una mostra $\mathcal{S} = \{w_1, w_2, \ldots, w_N\}$ es calcula, finalment, com la mitjana dels WERs de cada paraula:

$$WER_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{N} WER_{w_k}$$

3.6.1 Consideracions sobre el WER en WAs i PFAs

Ens centrem en la primera de les opcions: $\sigma_k = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\mathbf{Pr}\left(x_1 \cdots x_{k-1}\sigma\Sigma^*\right)\right)$.

Si mitjançant el mètode espectral hem aprés $f(x) = \mathbf{Pr}\left(x\Sigma^*\right)$, obtenim un autòmat $\mathcal{A} = \left\langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \right\rangle$, amb el qual la regla de predicció queda:

$$\sigma_k = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\mathbf{Pr}\left(x_1 \cdots x_{k-1}\sigma\Sigma^*\right)\right) = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\alpha_0^\top A_{x_1 \cdots x_{k-1}} A_\sigma \alpha_\infty\right)$$

Tanmateix, cal tenir en compte que si volem avaluar aquesta regla de predicció per a un WA $\mathcal{B} = \left\langle \beta_0, \beta_\infty, \{B_\sigma\}_{\sigma\in\Sigma} \right\rangle$ que computa una paraula sobre *strings* $f_{\mathcal{B}}$, com per exemple un PFA en què $f_{\mathcal{B}} = \mathbf{Pr}_{\mathcal{B}}$, l'avaluació d'aquesta mateixa regla de predicció és diferent. Si denotem per τ_k el símbol predit a la posició k-èsima amb l'autòmat $\mathcal{B}$:

$$\tau_k = \underset{\tau \in \Sigma}{\operatorname{argmax}} \left(\mathbf{Pr}\left(x_1 \cdots x_{k-1}\tau\Sigma^*\right)\right) = \underset{\tau \in \Sigma}{\operatorname{argmax}} \left[\sum_{i=1}^{n} \left(\beta_0^\top B_{x_1 \cdots x_{k-1}} B_\tau\right)_i\right]$$

Això és perquè la funció computada per $\mathcal{B}$ és $f_{\mathcal{B}}(x) = \mathbf{Pr}_{\mathcal{B}}\left(x\right)$ i no $\mathbf{Pr}_{\mathcal{B}}\left(x\Sigma^*\right)$.

Amb aquesta concepció del WER d'una màquina i amb aquesta regla de predicció, aquesta mètrica d'avaluació es fa idònia per a l'avaluació de màquines que aprenen sobre prefixos.

3.6.2 Consideracions sobre l'eficiència en el càlcul del WER d'una paraula

Per a WA apresos En el càlcul de $\sigma_k = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\alpha_0^\top A_{x_1 \cdots x_{k-1}} A_\sigma \alpha_\infty\right)$, el producte $A_\sigma \alpha_\infty$ és un vector que només depén de σi de α_∞, i és per tant constant per a cada símbol de cada paraula. Es pot, doncs, precalcular: $u_\sigma = A_\sigma \alpha_\infty$. Així, el càlcul de

$$\sigma_k = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\alpha_0^\top A_{x_1 \cdots x_{k-1}} A_\sigma \alpha_\infty\right) = \underset{\sigma \in \Sigma}{\operatorname{argmax}} \left(\alpha_0^\top A_{x_1 \cdots x_{k-1}} u_\sigma\right)$$

és substancialment més lleuger.

Per a PFA Notem l'igualtat:

$$\tau_k = \underset{\tau \in \Sigma}{\operatorname{argmax}} \left[\sum_{i=1}^{n} \left(\beta_0^\top B_{x_1 \cdots x_{k-1}} B_\tau\right)_i\right] = \underset{\tau \in \Sigma}{\operatorname{argmax}} \left[\beta_0^\top B_{x_1 \cdots x_{k-1}} B_\tau \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}\right]$$

En aquest cas podem precalcular

$$v_\tau = B_\tau \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

i realitzar la predicció mitjançant:

$$\tau_k = \underset{\tau \in \Sigma}{\operatorname{argmax}} \left[\sum_{i=1}^{n} \left(\beta_0^\top B_{x_1 \cdots x_{k-1}} B_\tau\right)_i\right] = \underset{\tau \in \Sigma}{\operatorname{argmax}} \left[\beta_0^\top B_{x_1 \cdots x_{k-1}} v_\tau\right]$$

3.7 Avaluació per *perplexity*

La perplexitat o *perplexity* és una mètrica utilitzada en teoria de la informació per a la comparació de models probabilístics. Es basa en la comparació de la distribució de probabilitat donada sobre un conjunt de paraules de test, entre el model real i la màquina hipòtesi. Aquesta comparació es fonamenta en el càlcul de l'entropia creuada entre ambdues distribucions de probabilitat.

L'avaluació per *perplexity* comprèn dues fases: la puntuació d'un conjunt de test i el càlcul de la *perplexity* en si.

En aquest apartat descrivim aquesta metodologia d'avaluació, per a la qual cosa farem servir la següent notació:

- Màquina *target* o objectiu: $\mathcal{T}$
- Màquina candidata o hipòtesi: $\mathcal{C}$
- Train set o conjunt d'entrenament: $\mathcal{U}$
- Test set o conjunt de test: $\mathcal{V}$

3.7.1 Puntuació del test set

Es proporciona un *training set* $\mathcal{U}$ generat per $\mathcal{T}$. Amb aquest training set, s'aprén una màquina candidata $\mathcal{C}$. Amb $\mathcal{C}$, s'avalua un conjunt de prova $\mathcal{V}$ i s'assigna una puntuació a cada paraula $w \in \mathcal{V}$.

Amb això, s'obté un vector de puntuacions $P_{\mathcal{V},\mathcal{C}}$, de mida $|\mathcal{T}|$. L'objectiu és que aquest vector assenyali, per a cada posició $(P_{\mathcal{V},\mathcal{C}})_i$, la probabilitat que la paraula w_i sigui generada d'entre les $|\mathcal{T}|$ paraules del *test set*. Per això, cal:

- $(P_{\mathcal{V},\mathcal{C}})_i > 0 \quad \forall w_i \in \mathcal{V}$
- $\sum_{w_i \in \mathcal{V}} (P_{\mathcal{V},\mathcal{C}})_i = \|P_{\mathcal{V},\mathcal{C}}\|_1 = 1$

Tractament dels negatius La màquina generada $\mathcal{C} = \langle \gamma_0, \gamma_\infty, \{C_\sigma\}_{\sigma\in\Sigma} \rangle$ no assegura que els valors $(P_{\mathcal{V},\mathcal{C}})_i$ siguin positius. Ha estat apresa mitjançant el mètode espectral i això no ens dóna cap garantia que els valors $\gamma_0^\top C_{w_i} \gamma_\infty$ siguin positius. En particular, el mètode espectral genera un *weighted automata* $\mathcal{C}$ que no serà, en general, un autòmat probabilista.

Cal, doncs, definir un tractament per aquells valors $(P_{\mathcal{V},\mathcal{C}})_i < 0$. Els dos tractaments més bàsics per a aquest cas són els següents:

- Prendre el valor 0: $\widehat{(P_{\mathcal{V},\mathcal{C}})_i} = 0$. A priori, l'opció més intuïtiva. Si una paraula ha de tenir *probabilitat* negativa, assignem-li probabilitat nul·la.
- Prendre el valor absolut: $\widehat{(P_{\mathcal{V},\mathcal{C}})_i} = \left|(P_{\mathcal{V},\mathcal{C}})_i\right|$. Opció no tan intuïtiva, però que proporciona millors resultats empíricament.

Per al cas particular d'avaluació per perplexitat o *perplexity*, l'opció escollida sempre serà prendre el valor absolut, ja que en la fórmula de la perplexity es pren el logaritme dels $(P_{\mathcal{V},\mathcal{C}})_i$, que no estaria definit si una paraula del *test set* tingués probabilitat nul·la.

Normalització Per aconseguir $\sum_{w_i \in \mathcal{V}} (P_{\mathcal{V},\mathcal{C}})_i = \|P_{\mathcal{V},\mathcal{C}}\|_1 = 1$, cal dividir el vector $P_{\mathcal{V},\mathcal{C}}$ per la seva norma 1: $\widehat{P_{\mathcal{V},\mathcal{C}}} = \dfrac{P_{\mathcal{V},\mathcal{C}}}{\|P_{\mathcal{V},\mathcal{C}}\|_1}$.

Així doncs, s'assigna a cada paraula $w_i \in \mathcal{V}$ una probabilitat $\mathbf{Pr}_{\mathcal{C}}(w_i) = \dfrac{\left|(P_{\mathcal{V},\mathcal{C}})_i\right|}{\|P_{\mathcal{V},\mathcal{C}}\|_1}$

3.7.2 Càlcul de la perplexitat

De la mateixa manera, s'avalua el conjunt de prova $\mathcal{V}$ amb l'autòmat *target* $\mathcal{T}$ i s'obté una probabilitat $\mathbf{Pr}_{\mathcal{T}}(w_i)$ per a cada $w_i \in \mathcal{V}$. Tenim, doncs, dues funcions:

$$\mathbf{Pr}_{\mathcal{T}}, \mathbf{Pr}_{\mathcal{C}} : \mathcal{V} \longrightarrow [0,1]$$

Definim la perplexitat mitjançant la següent fórmula:

$$perp(\mathcal{V}, \mathcal{T}, \mathcal{C}) = 2^{-\sum_{w \in \mathcal{V}} [\mathbf{Pr}_{\mathcal{T}}(w) \cdot \log_2(\mathbf{Pr}_{\mathcal{C}}(w))]}$$

En l'expressió de la *perplexity*, l'exponent és l'*entropia creuada* de les distribucions $\mathbf{Pr}_{\mathcal{T}}$ i $\mathbf{Pr}_{\mathcal{C}}$. És teorema que l'entropia creuada entre dues distribucions $\mathcal{D}$ i $\widetilde{\mathcal{D}}$ és mínima quan $\widetilde{\mathcal{D}} = \mathcal{D}$, per la qual cosa la perplexitat mínima s'assoleix quan $\mathcal{C} = \mathcal{T}$.

La naturalesa de la *perplexity* la fa adequada per a l'avaluació de màquines d'estats apreses sobre paraules senceres (*strings*), i poc adient per a màquines que aprenen sobre conjunts de prefixos.

D'altra banda, l'avaluació per perplexity requereix tenir dades del model real que no sempre estaran disponibles.

4 Mètode espectral: l'algorisme

En aquesta secció descriurem detalladament l'algorisme del mètode espectral. Com ja hem comentat anteriorment, podem dividir-lo en tres fases: preprocessament, nucli del mètode espectral i postprocessament o avaluació.

4.1 Preprocessament

La primera fase, el preprocessament, comprèn, d'una banda, l'obtenció d'una mostra de paraules del llenguatge generat per l'autòmat target. El model objectiu pot ser conegut (amb l'objectiu d'avaluar l'aprenentatge proporcionat pel mètode espectral), desconegut, o fins i tot tractar-se d'un model basat en dades del món real no necessàriament produïdes per un autòmat.

D'altra banda, el preprocessat comprèn l'adaptació de les dades disponibles als objectes matemàtics sobre els quals s'aplica el mètode espectral. Així, de la mostra n'extraiem informació que emmagatzemem en forma de matriu, i dóna lloc a les *matrius de Hankel* del model.

Per últim, el preprocessat considera també la millora de la informació emmagatzemada a les matrius de Hankel per mitjà de tècniques com l'*smoothing* o la normalització de la variància del subespai.

1. **Obtenció d'una mostra**

   Obtenció d'una mostra $\mathcal{M}$ del model que es pot aprendre. Aquesta mostra pot tenir diferents mides i diferent naturalesa: paraules d'un idioma, *logs* de pàgines web, seqüències d'ADN, ones sonores...

2. **Pretractament de la mostra**

   Qualsevol que sigui la naturalesa de la mostra, s'ha de traduir a un mateix format per poder ser tractat informàticament. Si l'alfabet és de n símbols, el representarem amb els enters de 0 a $n-1$, i les paraules seran seqüències d'enters. Això ens permet tractar qualsevol tipus de mostra en qualsevol tipus d'alfabet.

3. **Obtenció d'una base**

   Donada una mostra de mida N, $\mathcal{M} = \{w_1, w_2, \ldots, w_N\}$, cal trobar la matriu de Hankel de la mostra. Aquesta matriu de Hankel H serà una aproximació a la matriu de Hankel real $\widetilde{H}$ de la funció computada per l'autòmat target.

   Si bé l'ideal seria tenir la matriu de Hankel completa, computacionalment això és inviable, de manera que cal seleccionar una base. Disposarem, doncs, d'un sub-bloc de la matriu de Hankel. Per simplicitat, anomenarem directament *matriu de Hankel* aquest sub-bloc, i el denotarem per H. Anàlogament, denotarem per $\widetilde{H}$ el sub-bloc corresponent de la matriu de Hankel real.

   Si bé l'ideal seria tenir la matriu de Hankel completa, computacionalment això és inviable, de manera que cal seleccionar una base adequada. Són exemples de bases les següents:

   Base trivial La formada per $\mathcal{P} = \mathcal{S} = \Sigma \cup \{\lambda\}$.

   La base més senzilla, formada per cadascun dels símbols de l'alfabet. En cap cas garanteix que el rang de la matriu de Hankel en aquesta base sigui màxim. La informació que podrà emmagatzemar serà molt limitada i, per tant, l'autòmat que se n'obtingui tindrà molt poques garanties.

   Base aleatòria Obtinguda seguint criteris en què l'atzar tingui un paper destacat. Per exemple, a partir del següent algorisme:

```
inicialitzar P = S = ∅
for (w_i ∈ M) {
   escollir t enter, uniformement aleatori a 0 ≤ t ≤ |w_i|
   escriure w_i = u_i v_i, amb |u_i| = t i |v_i| = |w_i| − t
   afegir u_i a P i v_i a S
}
```

Amb alta probabilitat aquesta base garantirà la condició de rang, sempre i quan la mostra sigui prou completa. És possible que amb algun algorisme aleatori de construcció de bases obtinguem bases amb informació suficient per obtenir un autòmat vàlid. Ara bé, l'algorisme que hem presentat aquí té l'inconvenient de crear una base de mida $|\mathcal{M}|$, massa extensa i inviable computacionalment, tot i que això dependrà de la mostra.

Tanmateix, hauríem de procurar trobar una base que ens permeti obtenir una matriu de Hankel amb el màxim d'informació possible. És per això que a continuació proposem un algorisme orientat a maximitzar el nombre de *hits* a la matriu de Hankel.

Base de la màxima freqüència Formalment, si $f : \Sigma^* \longrightarrow \mathbb{R}$ és la funció a aprendre, ens interessa maximitzar el nombre d'elements del conjunt:

$$\{(u, v, w_i) \mid \quad u \in \mathcal{P}, v \in \mathcal{S}, w_i \in \mathcal{M}, f(uv) = w_i\}$$

És a dir, el que voldrem és trobar aquells conjunts $\mathcal{P}, \mathcal{S}$ que continguin els prefixos i sufixos més freqüents. Entenem els conceptes de *prefix* i *sufix* en un sentit més ampli que el purament lingüístic. En el nostre cas, parlarem de prefix u i sufix v sempre que ens referim a elements $u \in \mathcal{P}$ i $v \in \mathcal{S}$, o quan vulguem calcular el valor que pren la funció $f(uv)$. En aquest cas, quan pugui haver-hi confusió, en direm prefixos i sufixos *en el sentit de la matriu de Hankel*.

Aquesta visió de la base implica que l'algorisme per escollir-la variï segons quina sigui la funció f que vulguem aprendre amb el mètode espectral.

Amb alta probabilitat aquesta base garantirà la condició de rang, ja que se n'ha orientat l'obtenció a maximitzar el nombre de *hits* en la matriu de Hankel. En aquest sentit, considerem que aquesta base serà la que ens proporcioni la màxima informació possible d'entre les proposades, i per tant la que ens permetrà obtenir millors resultats.

Aprenent sobre *strings* La funció a aprendre és $f(w) = \mathbf{Pr}(w)$. Tant els prefixos com els sufixos en el sentit de la matriu de Hankel són els de la paraula w en el sentit lingüístic, és a dir, $\forall u, v \in \Sigma^* | w = uv$, u és un prefix i v és un sufix.

En aquest cas, la freqüència d'un prefix u és:

$$Freq_0(u) = \frac{1}{|\mathcal{M}|} \sum_1^N \mathbf{1}_{\{u \sqsubseteq_0 w_i\}}$$

I la freqüència d'un sufix v és:

$$Freq_\infty(v) = \frac{1}{|\mathcal{M}|} \sum_1^N \mathbf{1}_{\{v \sqsubseteq_\infty w_i\}}$$

Aprenent sobre *prefixos* La funció a aprendre és $f(w) = \mathbf{Pr}(w\Sigma^*)$. El concepte de prefix en el sentit de Hankel és el lingüístic, ja que el que és important d'un $u \in \mathcal{P}$ és que es maximitzin les probabilitats d'aparèixer al principi de la paraula.

Així doncs, la freqüència del prefix u torna a ser:

$$Freq_0(u) = \frac{1}{|\mathcal{M}|} \sum_1^N \mathbf{1}_{\{u \sqsubseteq_0 w_i\}}$$

El concepte de sufix, tanmateix, varia en aquest cas, i passa a ser el de *qualsevol substring que aparegui darrere d'un prefix*. Com que el que volem és maximitzar la probabilitat d'obtenir *hits* a la matriu de Hankel, i no coneixem a priori els prefixos de $\mathcal{P}$, la freqüència que tindrem en compte a l'hora d'escollir els elements de $\mathcal{S}$ és:

$$Freq_\infty(v) = \frac{1}{|\mathcal{M}|} \sum_1^N \mathbf{1}_{\{v \sqsubseteq w_i\}}$$

A l'hora d'escollir la *base de la màxima freqüència*, calcularem quins són els p prefixos i els s sufixos (en el sentit de Hankel) més freqüents de la mostra, essent p i s les mides desitjades per als conjunts $\mathcal{P}$ i $\mathcal{S}$.

Algorisme per calcular la base de la màxima freqüència Per calcular la base de la màxima freqüència, es proposa utilitzar l'estructura *map*, associant a claus de tipus *string* (els candidats a formar part de la base) valors de tipus enter (la freqüència absoluta de cadascun dels candidats). Proposem el següent algorisme per escollir els p prefixos més freqüents:

```
inicialitzar M com un map buit
for (w_i ∈ M) {
  per a cada prefix u de w_i (en el sentit de Hankel):
    si M t  definida la clau u, aleshores M[u]++
    altrament, crear la clau u a M i assignar-hi M[u]=1
}
escollir les claus de M que tenen els p valors m s grans
```

L'algorisme és anàleg per escollir els s sufixos més freqüents.

Discutim la implementació d'aquest algorisme en la Secció 5.2.1.

4. **Càlcul de les matrius de Hankel en la base escollida**

   Considerem la mostra $\mathcal{M}$, així com la base $\mathcal{P}, \mathcal{S}$ obtinguda en el pas anterior.

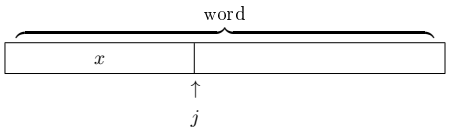
   En aquest pas cal calcular les matriu de Hankel de la mostra $\mathcal{M}$ en la base $\mathcal{P}, \mathcal{S}$. És a dir, cal recollir de la mostra tota la informació possible sobre f i emmagatzemar-la en la matriu de Hankel.

$$\begin{cases} H[u,v] &= f(uv) \\ H_\sigma[u,v] &= f(u\sigma v) \end{cases}$$

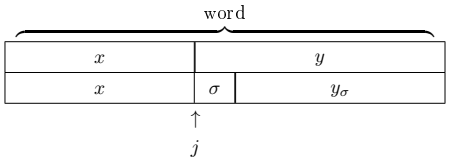
   A continuació descrivim, de manera esquemàtica, l'algorisme de càlcul de la matriu de Hankel segons la funció que es vulgui aprendre amb el mètode espectral. Es mostra cada paraula, així com les seves possibles particions, mitjançant una gràfica que pretén simplificar al màxim la comprensió de l'algorisme.

Aprenent sobre strings

```
inicialitzar H i les H_σ a 0
for (w_i ∈ M) {
  word = w_i
  for (j = 0; j <= word.size(); j++) {
    per a cada prefix x:
```



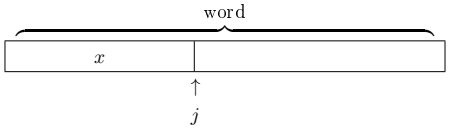

```
    if (x ∈ P) {
```



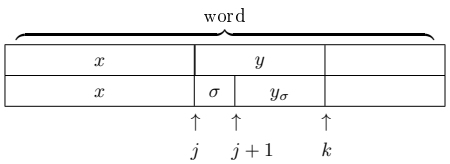

```
      if (y ∈ S)
        H[x][y]++
      if (y_σ ∈ S)
        H_σ[x][y_σ]++
}}}
```

Aprenent sobre prefixos

```
inicialitzar H i les H_σ a 0
for (w_i ∈ M) {
  word = w_i
  for (j = 0; j <= word.size(); j++) {
    per a cada prefix x:
```




```
    if (x ∈ P) {
      for (k = j; k <= word.size(); k++) {
```




```
        if (y ∈ S)
          H[x][y]++
        if (y_σ ∈ S)
          H_σ[x][y]++
}}}
```

4.2 Nucli del mètode espectral

El nucli del mètode espectral transforma les matrius de Hankel obtingudes en la fase de preprocessat en l'autòmat hipòtesi.

Aquesta fase conté la part més important de tot el procés, la *descomposició en valors singulars* o *singular value descomposition* (SVD). El mètode espectral està fonamentat en l'eficiència i la robustesa de l'SVD.

5. **Descomposició en valors singulars de la matriu de Hankel. Determinació del nombre d'estats**

   Considerem la matriu de Hankel H obtinguda en el pas anterior. Obtenim la descomposició en valors singulars de la matriu, $H = U\Lambda V^\top$.

   El nombre d'estats de la màquina *target* és, almenys, el de la seva matriu de Hankel real $\widetilde{H}$. Per tant, el nombre de valors singulars no nuls de la matriu Λ hauria de determinar el nombre d'estats de la nostra màquina hipòtesi.

   Ara bé, la matriu H obtinguda és només una aproximació a un sub-bloc de la matriu $\widetilde{H}$ obtinguda a partir d'una mostra, per la qual cosa está afectada per un factor d'error que amb alta probabilitat farà que el $rang(H)$ sigui màxim.

   Si l'error és suficientment petit (la qual cosa depén, en gran mesura, de la qualitat de la mostra i de la qualitat de la base escollida), hauria de ser senzill discernir quins dels valors singulars no nuls són realment nuls. Per exemple, determinant un llindar a partir del qual tots els valors més petits són negligibles i poden considerar-se nuls. Tanmateix, el que podria semblar fàcil havent fet aquesta observació és en realitat bastant més complex.

   En tot cas, sigui n el nombre d'estats escollit per a la màquina hipòtesi, ja sigui perquè s'ha inferit de la informació proporcionada per la mostra (a partir de l'anàlisi de la matriu Λ) o perquè s'ha fixat un valor que es consideri raonable.

6. **Construcció de l'autòmat hipòtesi**

   Siguin $H, \{H_\sigma, \sigma \in \Sigma\}$ les matrius de Hankel i n el nombre d'estats escollit per a la màquina hipòtesi. Sigui $H = U\Lambda V^\top$ una descomposició en valors singulars de H. Com ja s'ha esmentat a la Secció 3.5, construïm la màquina hipòtesi $\mathcal{A} = \langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \rangle$ de la següent manera:

   - $\alpha_0 = H[\lambda, \cdot] \cdot V_n$
   - $\alpha_\infty = (H \cdot V_n)^+ \cdot H[\cdot, \lambda]$
   - $A_\sigma = (H \cdot V_n)^+ \cdot H_\sigma \cdot H[\cdot, \lambda], \quad \forall \sigma \in \Sigma$

   On denotem per V_n la matriu formada per les n primeres columnes de V, i per $(H \cdot V_n)^+$ la pseudo-inversa de la matriu $H \cdot V_n$.

4.3 Avaluació

7. **Avaluació de l'autòmat hipòtesi**

Un cop obtingut un autòmat hipòtesi, cal trobar un criteri per avaluar com de bo és l'autòmat trobat respecte de l'autòmat target.

En proposem, en principi, dos: la *perplexitat* o *perplexity*, una mètrica emprada en teoria de la informació per a la comparació de models probabilístics, i el *word error rate*, una mètrica utilitzada sovint per a l'avaluació de sistemes relacionats amb el processament del llenguatge natural.

Avaluació per *perplexity*

- Adequada per avaluar l'aprenentatge de models dels qual la màquina real és coneguda.
- Adequada per avaluar l'aprenentatge sobre *strings*

Avaluació per *word error rate*

- Alternativa per avaluar l'aprenentatge de models dels quals es desconeix, o per als quals no existeix, la màquina real.
- Adequada per avaluar l'aprenentatge sobre *prefixos*

5 Implementació en R del mètode espectral

R és un llenguatge de programació lliure i gratuït utilitzat àmpliament per al desenvolupament de programes estadístics i d'anàlisi de dades. Recentment s'han anat incorporant al repositori de R un bon grapat de paquets que implementen funcionalitats que no se cenyeixen simplement a l'estadística, sinó que fan un salt cap a l'àmbit del *machine learning*.

El producte resultant d'aquest projecte entra en aquest últim conjunt de paquets. Es tracta, doncs, un paquet lliurement distribuïble escrit en R que implementa el mètode espectral per a l'aprenentatge de màquines d'estats: *spectralLearning*.

5.1 Descripció del paquet

Al paquet *spectralLearning*, les paraules són vectors d'enters i cada enter representa un símbol de l'alfabet. Per a *spectralLearning*, doncs, dos alfabets de la mateixa mida són el mateix alfabet. Això permet representar alfabets de mida arbitràriament gran, només limitada per la capacitat computacional de la màquina.

El paquet ha de ser útil tant per a l'usuari més principiant com per a l'usuari més avançat. Cal que, d'una banda, hi hagi funcions molt genèriques que permetin que l'usuari obtingui un autòmat correcte amb només una comanda i, per l'altra banda, proporcioni funcions que permetin una configuració més acurada dels paràmetres d'aprenentatge de l'autòmat (nombre d'estats, mida de la base, mode d'aprenentatge).

En aquest sentit, dividim el conjunt de funcions disponibles al paquet *spectralLearning* en diferents categories segons les seves funcionalitats:

1. Funcions bàsiques
2. Funcions avançades
3. Funcions avançades per a l'avaluació de resultats
4. Funcions avançades per a l'experimentació amb models en format PAutomaC
5. Altres funcions útils

La primera versió d'aquest paquet, si bé inclou totes les funcionalitats principals, no n'inclou algunes que podrien resultar interessants, com per exemple la detecció del nombre d'estats. Tampoc es fa un control d'errors en els paràmetres en la majoria de les funcions, cosa necessària en un llenguatge interpretat en què no es fan comprovacions de tipus.

El paquet porta inclosos els 48 *data sets* de la competició PAutomaC, per permetre a l'usuari experimentar i aprendre a utilitzar el paquet amb exemples. Per veure els *data sets* disponibles des de R es pot executar la comanda:

```
data(package="spectralLearning")
```

Els *data sets* disponibles són, per a XX entre 01 i 48:

pautomacsetXX Contenen tres objectes cadascun: `trainset` , `testset` i `solution` , amb conjunts d'entrenament i de prova i les puntuacions (probabilitats) assignades pel model real a les paraules del `testset` .

modelXX Conté l'objecte `model` , un model en el format PAutomaC (vegeu Secció 5.1.4).

Per carregar qualsevol dels *data sets*, només cal executar la comanda `data(dataset)` , per exemple, amb la comanda:

```
data(pautomacset01)
```

es carregaran al *workspace* de R els objectes `trainset` , `testset` i `solution` per al model n. 1 del PAutomaC. Cal tenir en compte que tant `trainset` com `testset` no són directament llistes de paraules, sinó dues llistes amb els atributs `wordList` i `alphSize` . Això és important alhora de cridar funcions d'aprenentatge sobre llistes de paraules com `spectral.automaton` .

Les distribucions dels paquets de R vénen amb una documentació en un format estàndard que pot generar-se automàticament a partir de directives indicades en el directori del paquet i comentaris inclosos al codi. Incloem aquesta documentació en el format estàndard de R a l'annex d'aquesta memòria.

A continuació fem una descripció breu de cadascuna de les diferents categories de funcions i de les funcions incloses en aquestes. Per a una descripció més detallada, vegeu l'annex.

5.1.1 Funcions bàsiques

En aquesta primera categoria aglutinem les tres funcions bàsiques del paquet, que permeten a l'usuari utilitzar el mètode espectral en la seva versió més senzilla, sense haver de preocupar-se especialment per la parametrització. En aquestes funcions, el *workflow* del mètode espectral (base, Hankel, autòmat) és completament transparent a l'usuari.

`spectral.automaton(word.list,alph.size,n.states,base.pref.size,base.suff.size,learning.mode)`
Retorna l'autòmat, après pel mètode espectral, per a la mostra `word.list` i amb els paràmetres indicats. Només `word.list` i `alph.size` són obligatoris, la resta de paràmetres són opcionals. `learning.mode` pot valer `prefixes` , `strings` o `substrings` .

`evaluate.by.perplexity(aut,word.list,solution.scores)`
Permet avaluar l'autòmat `aut` sobre el *test set* `word.list` mitjançant la *perplexity*, contra les puntuacions del model real `solution.scores` .

`evaluate.by.wer(aut,word.list)`
Permet avaluar l'autòmat `aut` sobre el *test set* `word.list` per *word error rate*.

5.1.2 Funcions avançades

Englobem en la categoria de funcions avançades aquelles funcions relacionades directament amb el mètode espectral que permeten a l'usuari la manipulació dels objectes que hi intervenen (base, matrius de Hankel, obtenció de l'autòmat).

Aquest conjunt de funcions ha de permetre l'usuari del paquet, per exemple, desenvolupar els seus propis algorismes per trobar aquells paràmetres que li són més adients.

`getBase(pref.size, suff.size, word.list, learning.type)`
Obté una base de prefixos i sufixos de la mida especificada, tenint en compte els que són més probables a `word.list` amb el mode d'aprenentatge especificat (base de la màxima freqüència).

`getHankelMatrices(alphSize,wordList,P,S,Hsigma, learning_type)`
Funció implementada íntegrament en C++ que obté les matrius de Hankel H i $H_\sigma \quad \forall \sigma \in \Sigma$, a partir de la llista de paraules (`wordList`) i les llistes de prefixos i sufixos (`P` i `S` , respectivament), i un mode d'aprenentatge.

`Hsigma` és un paràmetre únicament de sortida que ha d'estar prèviament inicialitzat per la funció `initialize.hankel.sigma` , la tasca de la qual és només reservar la memòria necessària per a les matrius H_σ. Això és perquè la llibreria Rcpp no permet el retorn de llistes amb un nombre arbitrari

d'elements. D'aquesta manera permetem l'obtenció de les H_σ a través d'un paràmetre de sortida passat per referència.

Noteu que, tot i que a R tots els paràmetres són passats per còpia, en tractar-se d'una funció implementada íntegrament en C++ podem utilitzar els paràmetres per referència i recuperar informació a través d'ells.

`initialize.hankel.sigma(alph.size,base.pref.size,base.suff.size)`
Retorna una llista de `alph.size` matrius de `base.pref.size` files × `base.suff.size` columnes, per passar-la com a paràmetre a la funció `getHankelMatrices` .

`getAutomaton(n, alphSize, H, Hsigma)`
Retorna l'autòmat de `n` estats generat pel mètode espectral a partir de les matrius de Hankel `H` i `Hsigma` .

`substring.2.prefix.aut(aut)`
Transforma un autòmat que computa una funció sobre *substrings* a un autòmat que computa una funció sobre *prefixos*.

`substring.2.string.aut(aut)`
Transforma un autòmat que computa una funció sobre *substrings* a un autòmat que computa una funció sobre *strings*.

5.1.3 Funcions avançades per a l'avaluació de resultats

Aquesta categoria de funcions permet l'avaluació de resultats amb un nivell de detall més gran que el proporcionat per les funcions bàsiques. La funció `evaluate.by.perplexity` es pot reproduir amb les funcions `evaluateWords` i `perplexity` , mentre que la funció `wer.as.prefix.automaton` té la mateixa funcionalitat que `evaluate.by.wer` .

`evaluateWords(word.list,aut)`
Aplica l'autòmat a cadascuna de les paraules de la llista i retorna un vector amb els valors obtinguts. Si l'autòmat és probabilista, el resultat és la probabilitat de generar aquesta paraula amb l'autòmat.

Aquesta funció pot ser útil per calcular la *perplexity*, però cal tenir en compte que depenent de quin sigui l'autòmat poden obtenir-se valors negatius. A més a més, la *perplexity* requereix un vector de suma 1.

`perplexity(cand,sol)`
Calcula la *perplexity* entre dues distribucions de probabilitat. Noteu que `cand` i `sol` són vectors numèrics amb les probabilitats ja calculades (típicament mitjançant la funció `evaluateWords`). Els vectors `cand` i `sol` han de tenir suma 1.

`wer.as.prefix.automaton(word.list,aut)`
Calcula el *word error rate* d'un autòmat sobre una llista de paraules tenint en compte que l'autòmat computa una funció sobre prefixos.

5.1.4 Funcions per a l'experimentació amb models en format PAutomaC

Les funcions d'aquesta categoria proporcionen a l'usuari la possibilitat d'experimentar amb models PAutomaC o amb el mateix format. El format dels models PAutomaC en R és una llista amb 4 elements, ordenadament: les probabilitats inicials dels estats, les probabilitats finals dels estats, una matriu amb

la probabilitat de generar cada símbol des de cada estat i una llista amb les matrius de transició per a cada símbol.

`modelToAutomaton(model)`
Retorna l'autòmat $\mathcal{A} = \langle \alpha_0, \alpha_\infty, \{A_\sigma\}_{\sigma\in\Sigma} \rangle$, en format de llista amb atributs `a1` , `ainf` i `A` , associat a `model` , el model en format PAutomaC.

`getSampleFromModel(model,N)`
Obté una mostra de `N` paraules generada amb un model en format PAutomaC.

`wer.as.PFA(word.list,aut)`
Calcula el *word error rate* d'un autòmat sobre una llista de paraules tenint en compte que l'autòmat és probabilista.

5.1.5 Altres funcions útils

Aquesta última categoria engloba altres funcions que poden resultar útils a l'usuari. Aquestes funcions són les següents:

Funcions que permeten obtenir un model d'n-grames (per $n \leq 3$)

- `unigram.model(word.list)`
- `bigram.model(word.list)`
- `trigram.model(word.list)`

Funcions que permeten avaluar un model d'n-grames (per $n \leq 3$)

- `wer.unigram(word.list,unigram)`
- `wer.bigram(word.list,n.gram)`
- `wer.trigram(word.list,n.gram)`

Funció per a l'escriptura de paraules En el paquet *spectralLearning*, les paraules són vectors d'enters. Aquesta funció permet, a partir d'un alfabet `alph` (un vector amb la correspondència d'enters a símbols), obtenir una representació d'una paraula com a *string*.

- `print.word(alph,word,sym.sep)`

5.2 Limitacions de R. La llibreria Rcpp

R és un llenguatge interpretat i la primera limitació que ens trobem en fer el salt d'un llenguatge com C o C++ a un llenguatge com R és la ineficiència manifesta en segons quines instruccions. Per exemple, una instrucció `for` que suma els enters des de 1 fins a 10 milions triga uns 45 segons en una màquina determinada.

D'altra banda, R ofereix algunes alternatives precompilades que acceleren tasques comuns sobre vectors. Per exemple, la instrucció `sum(1:10000000)` obté el mateix resultat que el `for` anterior en menys de mig segon en la mateixa màquina.

Tanmateix, aquestes alternatives no arriben ni de bon tros a la potència d'un llenguatge compilat: un petit programa en C que executa la mateixa tasca triga, en el mateix ordinador, 15 centèssimes de segon. Aquesta és només una mostra de les limitacions de R en temes d'eficiència.

Bàsicament, en R està "prohibit" utilitzar bucles. L'alternativa sempre passa per *vectoritzar*, tècnica que consisteix en programar aquelles funcions que han d'iterar sobre els elements d'un vector de manera que puguin aplicar-se sobre un únic element d'un vector i, aleshores, "iterar" per tots els elements amb funcions *precompilades* com `sapply` i `lapply` , que *apliquen* la funció a cadascun dels elements del vector per separat. Posteriorment, si cal, es poden combinar els resultats mitjançant funcions prefabricades com `sum` o `mean` .

De vegades, però, la vectorització no és suficient i cal recórrer a la programació d'algunes peces de codi en altres llenguatges més eficients i incloure-les convenientment en el codi R. En aquest sentit es publica el 2009 la llibreria *Rcpp*, fruit de la revisió d'un projecte embrionari anterior publicat el 2005.

La llibreria Rcpp permet l'execució en R de codi C++ prèviament compilat, facilitant enormement l'adaptació del codi C++ als tipus de dades existents en R. Tot i ser Rcpp un projecte recent i no tenir totes les funcionalitats desitjables (com ara el retorn cap a R d'una llista amb un nombre arbitrari d'elements), aquesta llibreria ha permès que *spectralLearning* sigui un paquet que, aplicant el mètode espectral sobre una mostra, obtingui un autòmat en qüestió de segons.

Ara bé, el C++ no fa miracles i serà igualment lent si l'algorisme no és prou eficient. En el proper apartat expliquem l'estructura de dades *vectorMap*, que ha permés reduir el temps d'obtenció d'una base en un 95%.

5.2.1 L'estructura *vectorMap*

A la Secció 4.1 hem descrit l'algorisme per al càlcul d'una base de la màxima freqüència. La implementació d'aquest algorisme en C++ directament (amb les adaptacions pròpies al llenguatge i a l'estructura de dades `map<vector<int>,int>`) té un comportament inesperadament lent.

Intentem explicar aquest comportament gràficament, suposant una implementació del `map` mitjançant un arbre binari de cerca *BST* senzill, tot i que la implementació més usual del `map` de la STL és l'estructura *red-black tree*, més complexa però tenint un BST com a base.

Suposem que s'han inserit en el `map` tots els prefixos per al següent conjunt de paraules:

$$\{abcab, aabca, bac, aabac, acba\} \subset \{a, b, c\}^*$$

En cada node, doncs, s'emmagatzema la freqüència del prefix que l'etiqueta. Mostrem una possible disposició del `map` resultant a la Figura 7. No hi mostrem els comptadors per a cadascun dels prefixos.

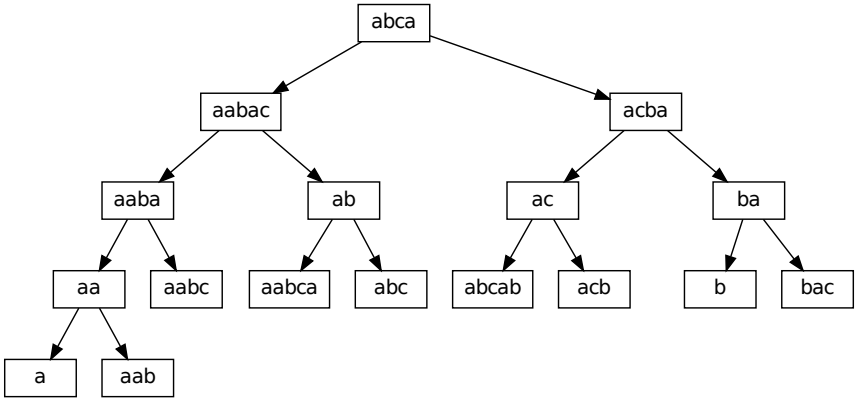


Figura 7: Disposició inicial del `map<vector<int>,int>`

Suposem ara que volem afegir la paraula *acbb*. Això és, cal afegir la informació relativa als prefixos

a, ac ,acb i $acbb$, ja sigui incrementant valors en entrades del `map` ja existents o creant-hi noves entrades. Mostrem el procés a la Figura 8.

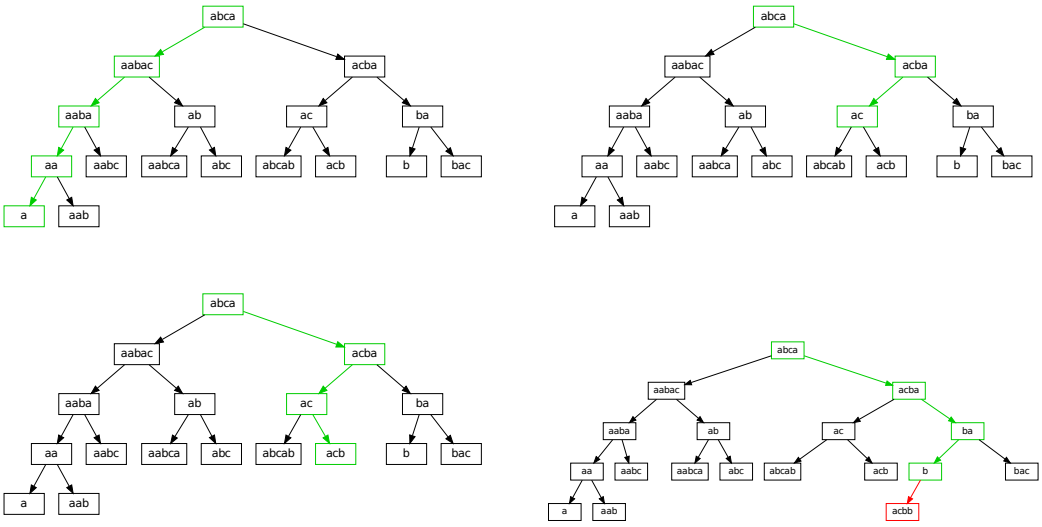


Figura 8: Inserció al `map<vector<int>,int>` dels prefixos de la paraula $acbb$

A través de la funció `find` de la classe `map` , s'ha d'iniciar una nova cerca per a cada prefix. A cada paraula s'ha de realitzar un mínim de comparacions per saber cap on ha de fer el següent pas. Per exemple, en el cas del prefix ac (Figura 8, superior dreta), calen:

- 2 comparacions a l'arrel, $a == a$ i $c > b$, per anar cap al fill dret.
- 3 comparacions al fill dret, $a == a$, $c == c$ i $longitud(ac) < longitud(acba)$, per anar cap al sub-fill esquerre.
- 3 comparacions al sub-fill esquerre, $a == a$, $c == c$ i $longitud(ac) == longitud(ac)$. Trobat!

Tenint en compte aquest nombre mínim de comparacions, construïm la següent llista:

a Recorre 5 claus, amb un mínim de $2 + 2 + 2 + 2 + 2 = 10$ comparacions.

ac Recorre 3 claus, amb un mínim de $2 + 3 + 3 = 8$ comparacions.

acb Recorre 4 claus, amb un mínim de $2 + 3 + 3 + 4 = 12$ comparacions.

$acbb$ Recorre 4 claus, amb un mínim de $2 + 4 + 1 + 1 = 8$ comparacions, i crea una clau nova.

En total, per a l'inserció dels prefixos del mot $acbb$ ha calgut recórrer 16 claus, fer 38 comparacions i crear una clau nova.

Ens plantegem trobar alguna manera d'optimitzar aquesta cerca, tenint en compte que, en cada paraula, cada prefix podria reaprofitar informació ja utilitzada per arribar al prefix anterior. En aquest sentit dissenyem l'estructura `VectorMap` en forma d'arbre n-ari, en què cada node conté un sol símbol i cada nivell (altura) representa la posició del símbol dins de la paraula.

En cada node del `VectorMap` , doncs, ja no s'emmagatzema la freqüència del prefix que l'etiqueta, sinó que el símbol que l'etiqueta només és l'últim símbol d'aquest prefix. Per recuperar tot el prefix caldria fer tot el recorregut fins a l'arrel.

Seguint amb l'exemple anterior, per al conjunt de paraules

$$\{abcab, aabca, bac, aabac, acba\} \subset \{a, b, c\}^*$$

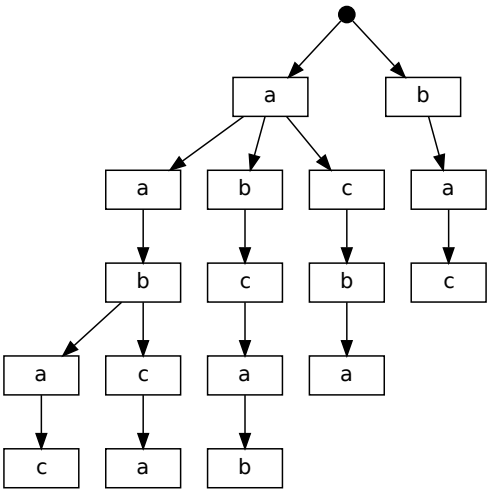


Figura 9: Disposició inicial del `VectorMap`

el `VectorMap` inicial és el que es mostra a la Figura 9.

En tractar-se d'un arbre n-ari, cada arbre, en comptes de tenir dos punters als nodes fills, té un `map<int,vectorMap*>` en què les claus són els símbols i els valors són punters als nodes fills. Per decidir el següent node, calen, doncs, un màxim de $\lceil \log_2 |\Sigma| \rceil$ comparacions, que serà el cost màxim de trobar un símbol en un `map` que tindrà, com a molt, $|\Sigma|$ claus.

Inserim ara, com en l'exemple anterior, la paraula *acbb*, és a dir, afegim una nova entrada per als prefixos *a*, *ac* ,*acb* i *acbb*, ja sigui incrementant valors en entrades del `map` ja existents o creant-hi noves entrades. Mostrem el nou procés a la Figura 10.

Amb aquesta estructura de dades, podem aprofitar la cerca de cada prefix per al prefix posterior. En aquest cas, el nombre de comparacions és el següent:

a Recorre 1 clau, amb un màxim de $\lceil \log_2 3 \rceil = 2$ comparacions.

ac Recorre 1 clau, amb un màxim de $\lceil \log_2 3 \rceil = 2$ comparacions.

acb Recorre 1 clau, amb un màxim de $\lceil \log_2 3 \rceil = 2$ comparacions.

acbb Recorre 1 clau, amb un màxim de $\lceil \log_2 3 \rceil = 2$ comparacions, i crea una clau nova.

En total, per a l'inserció dels prefixos del mot *acbb* en el `VectorMap` ha calgut recórrer 4 claus, fer com a molt 8 comparacions i crear una clau nova.

Es tracta només d'un exemple senzill i no és en cap cas una demostració rigorosa, pero queda bastant clar que el nombre de claus a recórrer i comparacions serà menor i que, per tant, el temps que es trigarà en insertar una mostra gran serà molt menor.

Com a exemple de temps de còmput, per a una mostra de 20000 paraules l'algorisme original trigava uns 50 segons i l'algorisme amb el `VectorMap` només triga uns 2 segons. Es tracta d'una reducció en temps del 95% que es manté per a diferents mostres i mides de mostres.

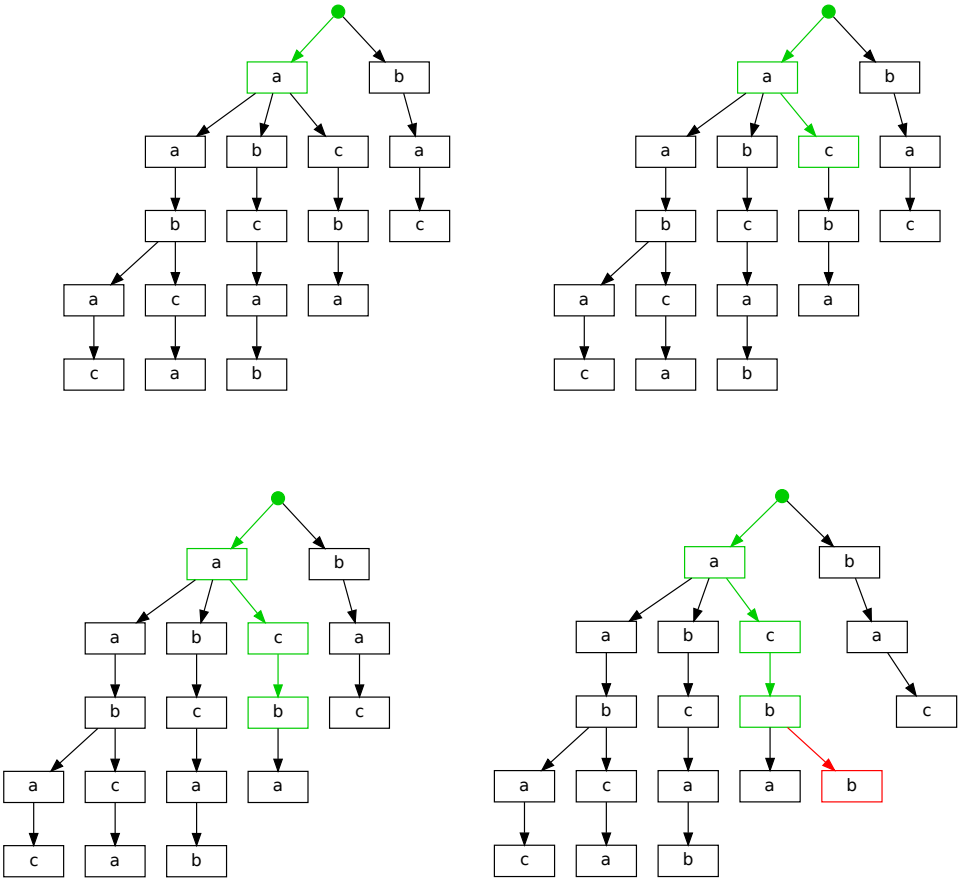


Figura 10: Inserció al `VectorMap` dels prefixos de la paraula *acbb*

6 Experimentació

En aquest capítol presentem un conjunt d'experiments i proves realitzats amb els paquets implementats. Aquesta experimentació ens permet, d'una banda, validar la implementació del mètode espectral en estar realitzant una prova integrada i massiva de tot el conjunt de funcions que conforma el paquet. D'altra banda, donem un primer conjunt de resultats obtinguts amb el mètode espectral com a mètode per a l'aprenentatge de la màquines d'estats.

En primer lloc presentem el conjunts d'autòmats amb els quals s'han realitzat els experiments. Es tracta dels models que van ser utilitzades en PAutomaC, una competició d'aprenentatge d'autòmats probabilístics que va tenir lloc l'any 2012.

També introduïm el model de trigrames, un model senzill però eficaç per a l'aprenentatge de llenguatges amb el qual comparem els resultats que obtenim amb el mètode espectral.

Per últim, descrivim i comentem els tres experiments que hem realitzat amb el mètode espectral i el conjunt de models PAutomaC.

6.1 El conjunt de proves: PAutomaC

PAutomaC va ser una competició d'aprenentatge d'autòmats probabilístics que es va dur a terme on-line durant l'any 2012. El seu objectiu era proporcionar una visió general de l'estat de la investigació en tècniques d'aprenentatge d'autòmats probabilístics, i comparar quins mètodes funcionen millor en quines condicions.

Els models utilitzats en la competició són públics. Els hem escollit com a conjunt de prova perquè proporcionen un marc ideal per a la validació del mètode espectral per a l'aprenentatge de màquines d'estats i de la seva implementació, ja que inclouen una varietat prou àmplia quant a tipus d'autòmats i permet comparar la correctesa del mètode en diferents situacions.

La competició consistia en l'aprenentatge de 48 models, generats aleatòriament en funció de quatre paràmetres i seguint algunes directrius.

Generació dels models PAutomaC La generació dels models ve condicionada pels següents paràmetres:

- El nombre d'estats, n, escollit aleatòriament amb $5 \leq n \leq 75$.
- La mida de l'alfabet, $|\Sigma|$, escollida aleatòriament amb $4 \leq n \leq 24$.
- La densitat de parelles estat-símbol que esdevenen transició, S_A, amb $0.2 \leq S_A \leq 0.8$.
- La densitat de transicions generades per a cada parella estat-símbol a un altre estat, S_T, amb $0 \leq S_T \leq 0.2$.

A partir d'aquests paràmetres, es generaven 16 models de cadascun dels tipus PFA, DPFA i HMM, seguint diferents directrius segons el tipus de model:

PFA Cada parella estat-símbol possible es genera amb una probabilitat S_A. Si la parella estat-símbol és generada, es genera una transició a un estat aleatori, creant així una terna estat-símbol-estat. Les ternes estat-símbol-estat restants es generen amb probabilitat S_T.

DPFA El mecanisme de generació és anàleg al dels PFA, però sense generar ternes estat-símbol-estat addicionals un cop generada la primera.

HMM Es generen primer parelles estat-estat amb probabilitat S_T. Després, per a cada parella estat-estat, cada símbol es pot emetre o no amb probabilitat S_A.

Finalment, sigui quin sigui el tipus de model, s'assignen les probabilitats a partir d'una distribució de Dirichlet. Això assigna les probabilitats per als estats inicials, per a l'emissió de símbols i per a les transicions.

Generació dels conjunts d'entrenament i prova Un cop generat el model, es genera un conjunt de prova de 1000 paraules, i un conjunt d'entrenament que tindrà longitud 100000 amb probabilitat 0.25, i longitud 20000 la resta de casos.

6.2 El model de trigrames

El model d'n-grames és un model que és sovint utilitzat en lingüística computacional per la seva senzillesa i pels seus bons resultats. Els n-grames més bàsics reben noms concrets: anomenem *unigrama* l'n-grama per a $n = 1$, *bigrama* l'n-grama per a $n = 2$ i *trigrama* l'n-grama per a $n = 3$.

Formalment, donat un alfabet Σ, un n-grama és una seqüència de n símbols d'aquest alfabet. Si considerem el conjunt de tots els n-grames Σ^n, el model d'n-grames defineix una distribució de probabilitat sobre aquest conjunt.

En particular, la distribució de probabilitat que defineix el model d'n-grames sobre Σ^n ens permet realitzar una predicció del següent símbol d'una paraula en funció dels $n-1$ símbols anteriors. Llegida una seqüència de símbols $\sigma_1\sigma_2\ldots\sigma_{k-1}$, utilitzarà la subseqüència $\sigma_{k-n+1}\ldots\sigma_{k-1}$ per decidir quin símbol predirà a la posició k-èssima de la seqüència. La determinació d'aquest símbol vindrà, generalment, donada pel símbol σ_k que maximitzi la freqüència de l'n-grama $\sigma_{k-n+1}\ldots\sigma_k$ en el conjunt d'entrenament. Per exemple, el model d'unigrames farà una predicció constant del símbol que sigui més freqüent en el *train set*.

Més concretament, el model d'n-grames utilitzarà models més bàsics si no disposa d'informació suficient. Així, utilitzarà sempre el model d'unigrames per predir el primer símbol d'una seqüència, el de bigrames per predir-ne el segon, i el de $(n-1)$-grames per al símbol $(n-1)$-èssim, fins a poder utilitzar el model d'n-grames en la resta de la seqüència. A més a més, si en algun moment la subseqüència dels $n-1$ símbols anteriors no existia en el *train set*, es prendran només els $n-2$ símbols anteriors, procedint anàlogament amb els $n-3$ i successius si no existís prou informació, fins a utilitzar el model d'unigrames si arribés a ser necessari.

En la mesura del que sigui computacionalment viable, aquesta n del model d'n-grames no estarà limitada, i s'utilitzarà sempre la totalitat de la seqüència llegida fins al moment.

Tanmateix, s'ha comprovat empíricament que amb el model de trigrames s'assoleixen resultats molt correctes, la millora dels quals no és molt rellevant si utilitzem n-grames amb $n > 3$, i és per això pel que l'utilitzarem en els nostres experiments per comparar els resultats amb els del model obtingut mitjançant el mètode espectral.

6.3 Experiment: Validació creuada

La validació creuada (en anglès, *cross-validation*) és una tècnica per a l'avaluació de models estadístics o predictius, encarada a garantir la independència dels resultats obtinguts respecte dels conjunts d'entrenament (*train sets*) i de prova (*test sets*) escollits.

Bàsicament, la *cross-validation* consisteix en particionar el conjunt de dades disponibles en diferents *train sets* i *test sets*, obtenir un resultat per a cadascuna de les particions i finalment *combinar* aquests resultats en un únic resultat final, generalment mitjançant la mitjana aritmètica.

La validació creuada és sovint utilitzada en models de predicció en què s'han de fixar alguns paràmetres per obtenir el model hipòtesi. Prenent com a únic conjunt de dades el conjunt d'entrenament, es realitza la *cross-validation* amb aquest *train set*, es determinen els paràmetres adients del model i finalment s'avalua amb les dades del conjunt de prova.

Aquest últim cas descrit és el nostre: El nombre d'estats de la màquina hipòtesi i la mida de la base escollida per generar la matriu de Hankel són dos paràmetres desconeguts però fonamentals en el mètode espectral. En aquest experiment, per a cada model del PAutomaC, farem servir la tècnica de la *cross-validation* en el *train set* per escollir els paràmetres a utilitzar per generar el model hipòtesi.

Un cop obtingut el model hipòtesi, es compararan els resultats amb el model real i el model de trigrames. S'utilitzaran com a mètriques d'avaluació tant el *word error rate* com la *perplexity*, però farem més èmfasi en la primera, ja que és la que ens permet comparar amb el model de trigrames. Recordem que la perplexitat no és adient per a l'avaluació d'n-grames degut a la seva naturalesa com a predictors de prefixos.

6.3.1 Objectius

- Utilitzar la tècnica de la *cross-validation* per determinar els valors òptims del nombre d'estats i la mida de la base.
- Comparar la màquina hipòtesi obtinguda amb el mètode espectral amb l'obtinguda amb el mètode de trigrames com a predictora de prefixos.
- Comparar la màquina hipòtesi obtinguda amb el mètode espectral amb altres alternatives utilitzades en la competició *PAutomaC*, per avaluar-la com a predictora d'*strings*.

6.3.2 Disseny experimental

S'utilitzarà la tècnica de validació creuada anomenada *de K iteracions* (en anglès, *K-fold cross-validation*, que consisteix en particionar el *train set* en K subconjunts i realitzar K iteracions del mètode espectral, utilitzant en cadascuna d'elles un dels subconjunts com a conjunt de prova i la unió de la resta dels subconjunts com a conjunt d'entrenament.

El nombre d'estats el farem variar des de 1 fins al doble del nombre d'estats de la màquina *target*. No considerem convenient provar més estats ja que el nombre d'estats que hauríem d'obtenir per a la màquina hipòtesi és el rang de la funció calculada pel *target*, que serà, de fet, menor o igual al seu nombre d'estats. Deixem, tanmateix, un marge de maniobra al mètode espectral per compensar possibles errors provinents de la mostra disponible. Limitarem el màxim a 80 estats perquè no es considera necessari augmentar aquest marge. Es recorrerà cadascun dels estats dins de l'interval, perquè en proves prèvies s'ha constatat que la variació en el resultat pot ser considerable.

La mida de la base la farem variar des de la mida de l'alfabet, $|\Sigma|$, a deu vegades aquesta mida, $10 \cdot |\Sigma|$, en intervals de $\frac{|\Sigma|}{2}$. Aquest pas de $\frac{|\Sigma|}{2}$ s'ha escollit perquè s'ha constatat en proves prèvies que petites variacions en la mida de la base no afecten substancialment el resultat. L'interval $[|\Sigma|, 10 \cdot |\Sigma|]$ s'ha escollit perquè, si bé a mesura que s'augmenta la base els resultats milloren, ho fan molt més lentament quan més gran ́se la mida de la base, mentre que el temps d'execució de l'algorisme es ressenteix notablement.

L'algorisme usat en l'experiment és, per a cada model, el següent:

1. Llegir $TS = trainset$ i $test = testset$ dels models PAutomaC.
2. Aprendre la màquina de trigrames amb TS
3. Calcular el WER obtingut amb la màquina de trigrames sobre *test*.
4. Per a cada n entre 1 i $min\{2 \cdot \#estats\_real, 80\}$
   - 4.1. Per a cada m entre $|\Sigma|$ i $10|\Sigma|$, de $0.5|\Sigma|$ en $0.5|\Sigma|$
     - 4.1.1. Per a cada i entre 1 i K
       i. $TS_i = c\left(TS\left[1 : \frac{|TS|}{K}\right], \cdots, \cancel{TS\left[(i-1)\frac{|TS|}{K} + 1 : i\frac{|TS|}{K}\right]}, \cdots, TS\left[(K-1)\frac{|TS|}{K} + 1 : K\right]\right)$

ii. Llençar mètode espectral (sobre prefixos) amb n estats i una base de mida m, amb *train set*: TS_i

iii. Calcular el WER W_i obtingut sobre el test set: $TS\left[(i-1)\frac{|TS|}{K} : i\frac{|TS|}{K}\right]$, amb la màquina apresa.

4.1.2. Calcular el WER obtingut de la cross-validation com la mitjana dels W_i: $W_{cv} = \frac{1}{K}\sum_{i=1}^{K} W_i$

5. Determinar valors de n i m òptims, n_{opt} i m_{opt}, en funció del millor dels W_{cv}.

6. Llençar mètode espectral (sobre prefixos) amb n_{opt} estats i una base de mida m_{opt}, amb train set: TS

7. Calcular el WER W obtingut sobre el test set *test*.

8. Llençar mètode espectral (sobre *strings*) amb n_{opt} estats i una base de mida m_{opt}, amb train set: TS

9. Calcular la perplexity obtinguda amb la màquina apresa.

6.3.3 Exposició i anàlisi de resultats

A la Taula 2 presentem un resum dels resultats obtinguts en la validació creuada, que comentem i analitzem fent ús de representacions gràfiques. Noteu que hi expressem els valors S_A i S_T en percentatges.

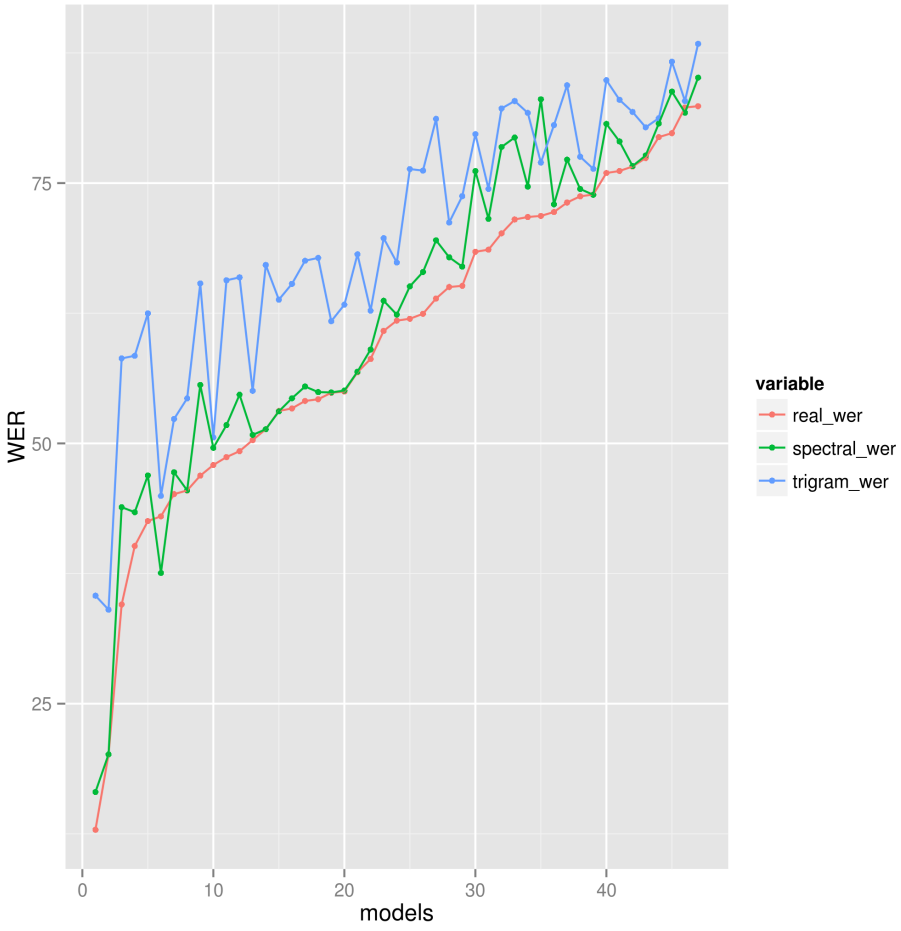


Figura 11: WER mètode espectral vs. trigrames i model real

	Paràmetres model						WER			*perplexity*		òptims	
ID	n	$\|\Sigma\|$	S_A	S_T	N	type	real	3-gram	spec	real	spec	n_{opt}	base
1	63	8	33	9	20k	Hmm	68.61	74.44	71.57	29.90	30.58	22	80
2	63	18	33	2	20k	Hmm	54.08	67.54	55.46	168.33	176.18	21	171
3	25	4	79	8	20k	Pfa	45.13	52.34	47.23	49.96	50.48	19	40
4	12	4	44	15	100k	Pfa	43.00	44.96	37.56	80.82	80.93	17	36
5	56	6	29	2	20k	Hmm	20.13	34.03	20.13	33.24	33.25	12	33
6	19	6	48	5	20k	Dpfa	47.92	50.58	49.58	66.98	67.53	18	60
7	12	13	24	8	20k	Dpfa	51.36	67.14	51.36	51.22	51.35	15	110
8	49	8	36	6	100k	Pfa	54.24	67.82	54.93	81.38	81.71	37	80
9	71	4	39	1	20k	Dpfa	12.89	35.38	16.52	20.84	43.28	18	38
10	49	11	63	2	20k	Pfa	63.91	81.17	69.50	33.30	34.09	46	110
11	47	20	49	2	20k	Dpfa	73.14	84.40	77.25	31.81	32.84	47	200
12	12	13	35	11	20k	Pfa	54.98	63.32	55.08	21.66	21.76	12	123
13	63	4	69	2	100k	Dpfa	34.53	58.16	43.87	62.81	77.53	27	40
14	15	12	49	8	20k	Hmm	65.02	71.21	67.88	116.79	117.52	7	60
15	26	14	41	7	20k	Pfa	60.81	69.71	63.70	44.24	45.01	26	140
16	49	10	62	2	100k	Dpfa	62.45	76.20	66.46	30.71	31.01	40	100
17	22	13	22	17	20k	Pfa	53.36	65.33	54.33	47.31	48.00	21	123
18	25	20	23	4	100k	Dpfa	53.11	63.80	53.07	57.33	58.58	23	160
19	68	7	33	4	100k	Hmm	48.68	65.66	51.75	17.88	17.95	28	70
20	11	18	39	15	20k	Hmm	73.73	77.53	74.43	90.97	94.31	7	180
21	56	23	25	5	20k	Hmm	71.85	76.97	83.05	30.52	36.96	62	230
23	33	7	38	11	100k	Hmm	58.11	62.75	59.01	18.41	18.45	19	70
24	6	5	50	17	20k	Dpfa	45.48	54.32	45.49	38.73	38.76	5	45
25	40	10	58	5	20k	Hmm	68.40	79.71	76.16	65.74	71.10	67	100
26	73	6	59	1	20k	Dpfa	46.90	65.36	55.61	80.74	88.40	41	60
27	19	17	64	5	20k	Dpfa	71.74	81.75	74.67	42.43	43.02	21	170
28	23	6	75	11	20k	Hmm	65.13	73.75	66.97	52.74	53.14	9	60
29	36	6	38	4	20k	Pfa	42.54	62.49	46.92	24.03	24.85	37	60
30	9	10	66	18	20k	Pfa	61.79	67.38	62.37	22.93	23.09	9	100
31	12	5	38	20	20k	Pfa	50.32	55.05	50.81	41.21	41.54	12	50
32	43	4	77	2	100k	Dpfa	40.15	58.42	43.39	32.61	32.87	28	40
33	13	15	59	12	20k	Hmm	72.22	80.57	72.96	31.87	32.40	3	127
34	64	21	37	3	20k	Pfa	70.18	82.17	78.47	19.95	20.97	61	210
35	47	20	36	2	20k	Dpfa	61.97	76.35	65.08	33.78	35.29	52	190
36	54	9	63	7	100k	Hmm	76.16	83.00	79.00	37.99	38.15	13	90
37	69	8	52	18	100k	Pfa	82.26	82.88	81.76	20.98	21.04	14	80
38	14	10	79	19	20k	Hmm	79.42	81.23	80.73	21.45	21.53	3	80
39	6	14	42	18	20k	Pfa	56.83	68.16	56.88	10.00	10.03	6	119
40	65	14	65	2	20k	Dpfa	71.51	82.88	79.38	8.20	8.33	54	140
41	54	7	69	14	100k	Hmm	77.40	80.36	77.65	13.91	13.94	9	70
42	6	9	52	17	20k	Dpfa	54.86	61.73	54.90	16.00	16.03	7	27
43	67	5	60	16	20k	Pfa	73.90	76.37	73.87	32.64	32.93	8	22
44	73	13	63	6	20k	Hmm	82.39	88.38	85.13	11.71	11.83	6	110
45	14	19	80	9	20k	Hmm	76.59	81.83	76.64	24.04	24.08	2	28
46	19	23	49	10	20k	Pfa	75.97	84.89	80.69	11.98	12.27	23	230
47	61	15	30	2	100k	Dpfa	49.25	65.95	54.68	4.12	4.15	61	150
48	16	23	70	6	20k	Dpfa	79.80	86.67	83.79	8.04	8.26	16	230

Taula 2: Resum de resultats de la *cross-validation*

A la Figura 11 s'han ordenat els models pel WER del model real i es mostra el WER obtingut per la màquina obtinguda amb el mètode espectral i per la màquina de trigrames.

A la figura s'aprecia clarament que el mètode espectral supera amb escreix el model de trigrames en termes de WER, essent només superat en un dels models (n. 21 a la Taula 2). La diferència es fa menys

notable a mesura que el *word error rate* del model real és més gran, la qual cosa es justifica pel fet que hi ha menys marge d'error: si un model té un WER del 40% pot succeir que una màquina apresa tingui un WER del 50% o del 60%, però no s'espera que per a un model amb un WER del 80% cap màquina apresa tingui un error proper al 100%.

També podem observar alguns models per als quals el mètode espectral supera el real en termes de WER (el cas més clar és el model n. 4 de la Taula 2, sisè a la gràfica, amb un alfabet de només 4 símbols), i hi ha un bon nombre de models en què els WERs de models real i espectral són pràcticament idèntics: és el cas dels models 24, 12 i 43, que ocupen les posicions 8, 20 i 39 a la gràfica, respectivament.

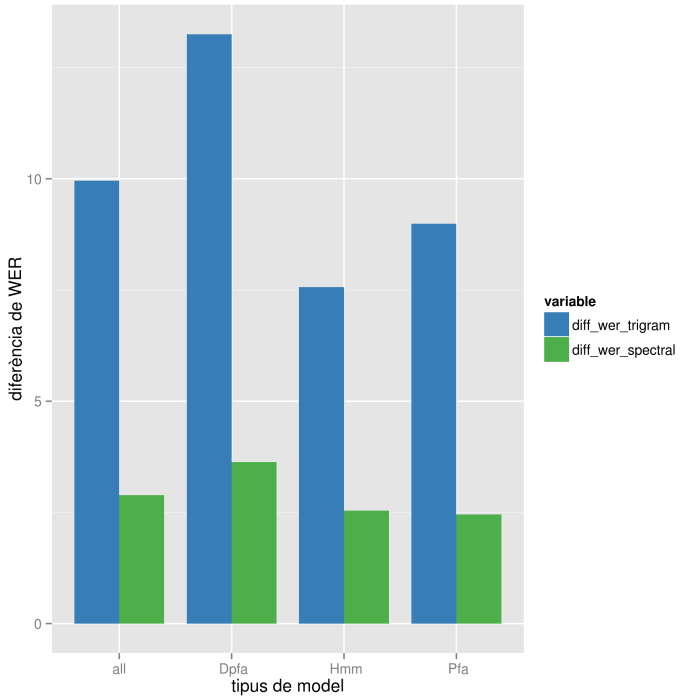


Figura 12: Diferència de WER mitjana segons tipus de model. Mètode espectral i trigrames

A la Figura 12 hem volgut mostrar una comparació del mètode espectral entre els tres diferents tipus d'autòmats considerats: DPFA, HMM i PFA. Per això, hem calculat la mitjana dels valors absoluts de les diferències de WER entre model espectral i model real per a cadascun dels tres tipus de màquina i ho hem mostrat en una gràfica de barres. Ho fem en termes de WER en una primera anàlisi per poder comparar així les diferències amb el model de trigrames. També mostrem la mitjana global de les diferències per a tots els models.

Observem primerament que, com ja hem constatat en la figura anterior, el model espectral és molt superior al mètode de trigrames en termes de WER.

Podem veure que el model més difícil de predir és el DPFA. Particularment en el model de trigrames, la diferència de WER per a DPFAs està més de tres punts per sobre de la mitjana. Aquest comportament és similar en el mètode espectral: la mitjana de diferències per a DPFAs està bastant per sobre de la mitjana de diferències per a tots els models.

Observem, però, una lleugera diferència en el comportament de trigrames i espectral: mentre que el model de trigrames es comporta força millor amb HMMs que amb PFAs, no és així en el cas del mètode espectral, en què la diferència de WER entre PFAs i HMMs s'equilibra. Fins i tot observem que són lleugerament millors els resultats de PFAs, tot i que no de manera rellevant ni estadísticament significativa.

En les Figures 13 i 14 avaluem el mètode espectral segons cadascun dels models utilitzant la *perplexity*.

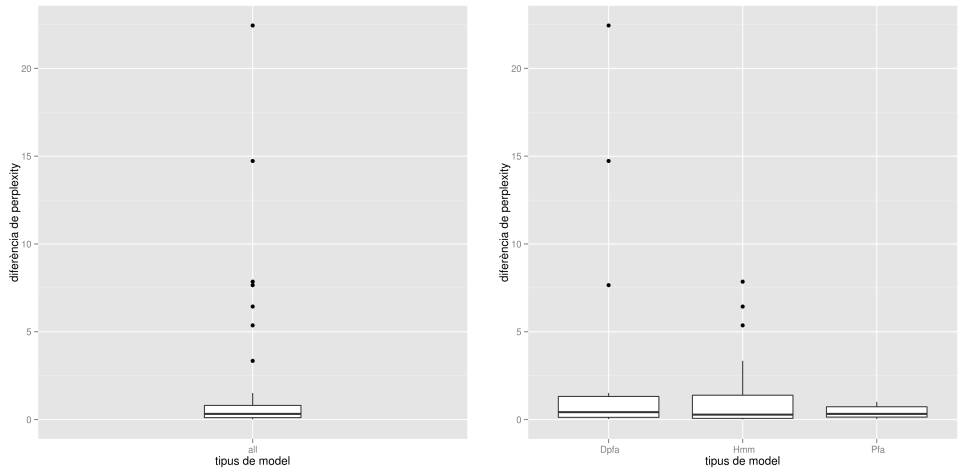


Figura 13: Diferència de *perplexity* amb el mètode espectral amb *outliers*

Concretament, mostrem mitjançant *box plots* la diferència de *perplexity* entre el model espectral i el model real.

A la Figura 13 hem mantingut els *outliers*, i podem veure que són inexistents per als PFA: el mètode espectral és un mètode molt fiable per aprendre PFAs. Més del 75% de valors estan per sota de l'1, la qual cosa ens permet afirmar que el mètode espectral funciona molt bé en general. La pitjor diferència de *perplexity* és de 22.4, i correspon a un model particular amb molts estats i un alfabet de només 4 símbols. Tot i ser un valor alt, no considerem que sigui excessivament dolent, ja que la *perplexity* és una mètrica que penalitza molt els errors i fàcilment tendeix a l'infinit.

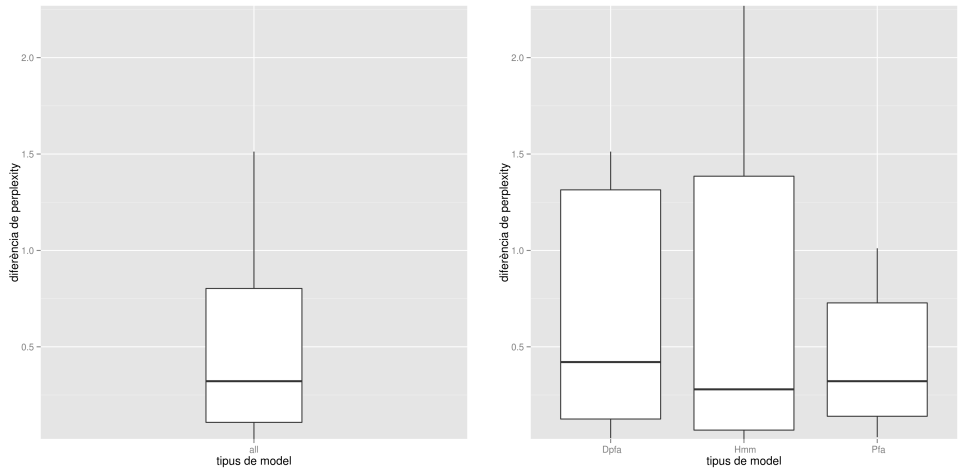


Figura 14: Diferència de *perplexity* amb el mètode espectral sense *outliers*

La Figura 14 és una rèplica de la Figura 13 en què hem fet *zoom* per apreciar millor la distribució de les diferències de *perplexity*.

Hi podem veure que la diferència de *perplexity* no només està per sota d'1 en un nombre considerable de casos, sinó que en més del 50% dels casos està per sota de 0.5. Això passa per a tots els tipus de models, la qual cosa ens permet afirmar que, si bé els models DPFA han estat els més difícils de predir, en general són predits d'una manera bastant correcta, però és més fàcil trobar un cas particular en què el mètode espectral no funcioni tan bé com voldríem.

Finalment, a la Figura 15 mostrem una comparació de les diferències de *perplexity* en funció de la grandària de la mostra del model PAutomaC. Recordem que un de cada quatre models tenia una mostra

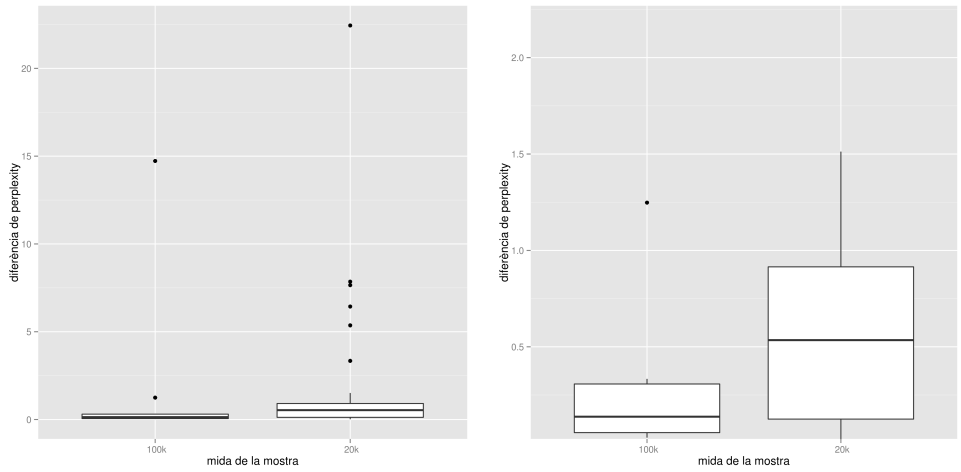


Figura 15: Diferència de perplexity amb el mètode espectral segons la mida de la mostra, amb i sense *outliers*

de 100000 paraules i el 75% restant una mostra de 20000 paraules.

Com era d'esperar, la *perplexity* dels resultats és millor en general si la mostra és més gran: podem observar que els resultats són més petits per a les mostres de 100000 paraules.

La corba d'aprenentatge dels models en funció de la mida de la mostra es tracta amb més detall en el següent apartat.

6.4 Experiment: Corba d'aprenentatge

En aquest experiment es pretèn avaluar la capacitat d'aprenentatge del mètode espectral a mesura que augmenta la mida de la mostra. L'objectiu és comprovar que, tot i que amb mostres petites es veu superat per models més simples, com el de trigrames, a mesura que la mostra és més gran el resultat s'apropa a la màquina objectiu.

Es vol també comparar la corba d'aprenentatge del mètode espectral dels diferents tipus d'autòmats presents en el conjunt del PAutomaC: HMM, PFA i DPFA.

6.4.1 Objectius

- Avaluar l'aprenentatge del mètode espectral segons la mida de la mostra.
- Avaluar l'aprenentatge del mètode espectral segons el tipus de model: HMM, PFA, DPFA.
- Comparar la corba d'aprenentatge del mètode espectral amb la del model de trigrames.

6.4.2 Disseny experimental

Per aquest experiment s'utilitzarà el paquet de *sampling*, que permet generar una mostra a partir d'un model. Per a cada mida de la mostra, es generaran 20 *train sets* i s'obtindrà una màquina hipòtesi per a cadascun d'aquests conjunts d'entrenament. Cada màquina hipòtesi s'avaluarà contra el *testset* del PAutomaC. Es realitzarà el nombre de 20 iteracions perquè és el que s'ha considerat òptim tenint en compte la relació entre quantitat d'informació obtinguda i temps d'execució.

La mida de la mostra s'incrementarà exponencialment, duplicant-la a cada iteració. Això ens permetrà avaluar el comportament del mètode espectral amb mostres relativament petites i amb mostres relativament grans, sense necessitat de passar per tots i cadascun dels valors intermitjos. L'interval on es mourà la mida de la mostra serà el $[1000, 512000]$, que comprèn les dues mides de mostra presents en els conjunts del PAutomaC (20000 i 100000).

S'escolliran els paràmetres del nombre d'estats i mida de la base òptims segons el resultat de la validació creuada de l'experiment anterior.

L'algorisme usat en l'experiment és, per a cada model, el següent:

1. Llegir resultats de l'experiment anterior: n i m òptims
2. Llegir $test = testset$ dels models PAutomaC (longitud: 1000 paraules).
3. $N = 1000$
4. Mentre $N \leq 512000$
   - 4.1. Per *it* entre 1 i 20
     - 4.1.1. Generar *train set* de N paraules (independents)
     - 4.1.2. Calcular el WER obtingut amb la màquina de trigrames
     - 4.1.3. Llençar mètode espectral (sobre prefixos) amb n estats i una base de mida m
     - 4.1.4. Calcular el WER obtingut amb la màquina apresa
   - 4.2. $N* = 2$

6.4.3 Exposició i anàlisi de resultats

Per aquest experiment mostrarem un subconjunt de les corbes d'aprenentatge dels models de PAutomaC. Per poder apreciar millor els resultats, aquests es mostren amb una escala logarítmica en l'eix de les abscisses.

Els punts de la corba corresponen a la mitjana de les 20 iteracions que s'han realitzat amb cada model. Per sobre i sota de la corba apareix una franja grisa mostrant els valors de WER màxim i mínim per a cadascuna de les mides de la mostra, amb la intenció de donar una idea de la variabilitat que poden arribar a tenir els resultats en funció de quina sigui la mostra.

El comportament esperat de les màquines apreses amb el mètode de trigrames és que, si bé a mesura que la mostra augmenta pot arribar a assolir millors resultats en termes de WER, aquesta millora no és especialment notable. De fet, el mínim valor de WER s'assoleix ben aviat. En canvi, s'espera que les màquines apreses mitjançant el model espectral puguin començar amb resultats dolents per a mostres petites, però que aquests resultats millorin substancialment en anar augmentant la mida de la mostra.

Aquest comportament és l'habitual en l'àmbit del *machine learning*. Els models d'aprenentatge més senzills solen tenir millors resultats que models més complexos en mostres més petites, però es veuen superats amb escreix a mesura que la mida de la mostra creix. Això es pot justificar pel fet que els models complexos necessiten més dades per acabar d'ajustar els major nombre de paràmetres que utilitzen per aprendre el model.

A la Figura 16 es mostra un cas ben clar d'aquest comportament. Mentre que el model de trigrames comença amb un WER del 83.5% per a mostres de 1000 paraules i amb mostres de 16000 ja ha assolit pràcticament el WER mínim, poc per sobre de la franja del 82%, el mètode espectral comença amb una predicció molt pitjor d'aquest model, però a partir de les 8000 paraules ja supera el model de trigrames. A més, segueix millorant fins a situar-se a tres punts percentuals per a la mostra més gran considerada en aquest experiment, de 512000 paraules, tot i que la corba fa pensar que podria arribar a aconseguir-se alguna millora més si se seguís augmentant la mida de la mostra.

Observem que els resultats obtinguts en l'experiment anterior es corresponen amb els d'aquest. En la *cross-validation* havíem obtingut un WER del 78.5% amb una mostra de 20000 paraules, la qual cosa és consistent amb la Figura 16, que mostra un WER mitjà d'un 79% per a una mostra de 16000 paraules i de poc més del 77% per a una mostra de 32000 paraules.

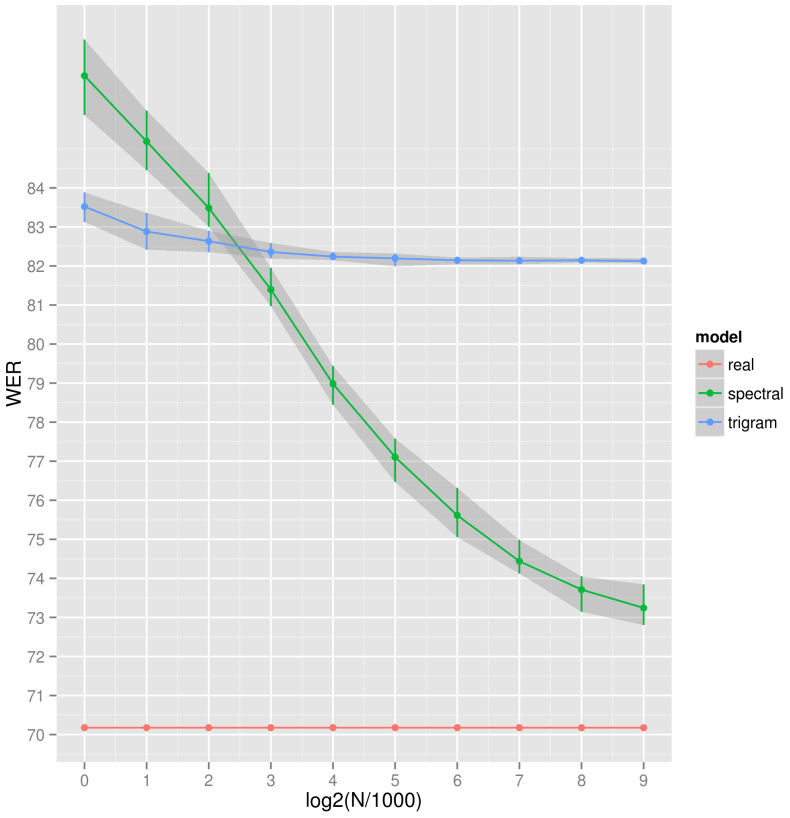


Figura 16: Corba d'aprenentatge per al model 34 (PFA)

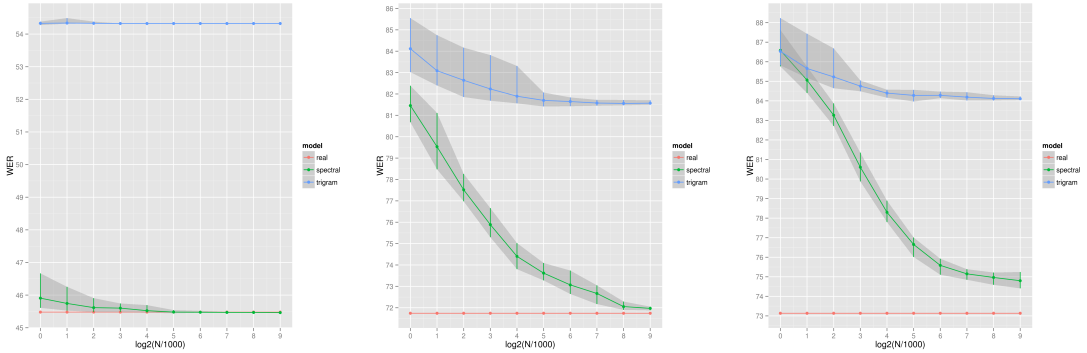


Figura 17: Corba d'aprenentatge per als models 24, 27 i 11 (DPFA, 6, 19 i 47 estats, resp.)

A la Figura 17 mostrem tres exemples de corbes d'aprenentatge per a mostres obtingudes de DPFAs. Hi podem observar novament, en primer lloc, que el model espectral és clarament superior al de trigrames, i ho és més encara a mesura que s'augmenta la mida de la mostra.

També veiem que el model de trigrames sembla més sensible a les variacions prenent mostres de la mateixa mida, ja que les franges al voltant de la corba són més amples.

Per últim, sembla que l'aprenentatge és més lent si el nombre d'estats del model real és més gran, tot i que no podem generalitzar aquesta afirmació a partir d'uns pocs models, ja que també existeix algun exemple que contradiu aquesta afirmació. Caldrien bastants més models, ja que 16 models de cada tipus són pocs per poder extreure conclusions en aquest aspecte si l'evidència no és del tot clara.

A la Figura 18 mostrem l'anàleg de la Figura 17 per a HMMs. El comportament és molt similar a l'observat en la figura anterior, si bé sembla que per a aquest tipus d'autòmats el mètode espectral sigui més sensible a les variacions en la mostra que per als DPFAs, ja que les franges grises al voltant de la

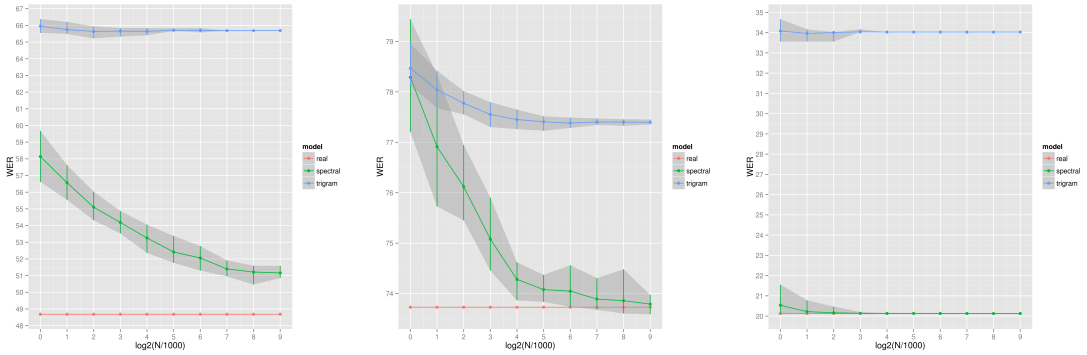


Figura 18: Corba d'aprenentatge per als models 19, 20 i 5 (HMM, 68, 11 i 56 estats, resp.)

corba són lleugerament més amples (tenint en compte les diferències d'escala en l'eix d'ordenades). Com hem comentat abans, no podem extreure conclusions generals d'uns pocs models particulars.

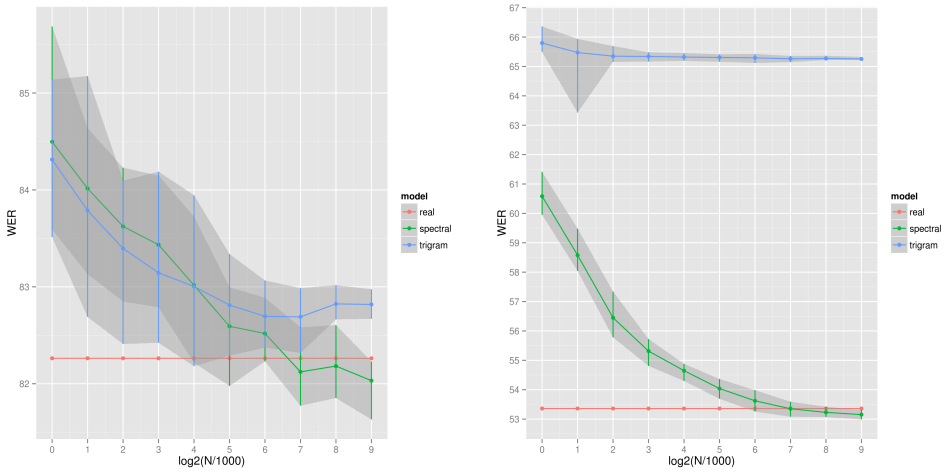


Figura 19: Corba d'aprenentatge per als models 37 i 17 (PFA, 69 i 22 estats, resp.)

Per últim, a la Figura 19 mostrem dos casos en què el mètode espectral arriba a superar el model real en termes de WER.

6.5 Experiment: Sensibilitat a la variació dels paràmetres

En aquest experiment es pretèn avaluar la sensibilitat del mètode espectral a la variació dels paràmetres més representatius. L'objectiu és descriure el comportament del mètode segons si la predicció del nombre d'estats o bé la mida escollida per a la base varia respecte de l'òptim.

Es vol també comparar la sensibilitat del mètode espectral en funció dels diferents tipus d'autòmats presents en el conjunt del PAutomaC: HMM, PFA i DPFA.

6.5.1 Objectius

- Descriure la sensibilitat del mètode espectral a la variació del nombre d'estats i la mida de la base.
- Comparar la sensibilitat als paràmetres segons el tipus d'autòmat.

6.5.2 Disseny experimental

Per a aquest experiment s'utilitzarà la informació recollida en l'experiment de la validació creuada, en què s'ha calculat el WER per a diferents valors del nombre d'estats i la mida de la base.

El *train set* utilitzat en cadascun dels casos ha estat el del conjunt de models PAutomaC, i la mètrica d'avaluació és el *word error rate* obtingut mitjançant una *10-fold cross-validation* sobre aquest *train set*.

6.5.3 Exposició i anàlisi de resultats

En aquest experiment representem els resultats mitjançant un mapa degradat de color, en una matriu amb el nombre d'estats a l'eix de les x i la mida de la base, en múltiples de la mida de l'alfabet $|\Sigma|$, a l'eix de les y. S'han assignat colors més calents (vermells, taronges) als valors més petits de WER i colors més freds (blaus, morats) als valors més alts.

Concretament, s'han assignat els colors entre el vermell i el blau fosc distribuïts proporcionalment entre aquells valors de WER situats per sota del tercer quartil, i des del sisè fins al setè colors en què es dispersa la llum s'ha distribuït el quartil restant. Amb això aconseguim visualitzar d'una manera més clara els canvis al voltant dels valors òptims de WER, que són els més interessants experimentalment.

A continuació mostrarem i comentarem breument un subconjunt dels *colormaps* obtinguts per als models de PAutomaC, diferenciant primer entre els tres tipus d'autòmats considerats i finalment fent èmfasi en dos casos més patològics que s'han observat.

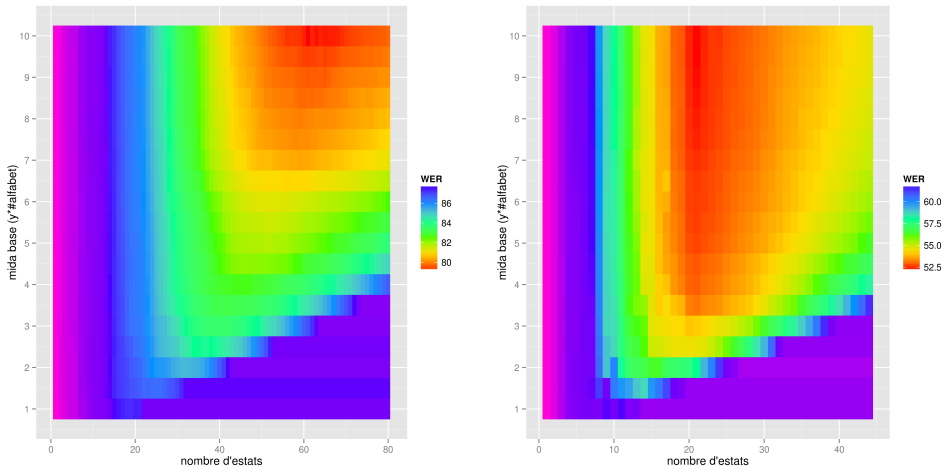


Figura 20: Sensibilitat a paràmetres per als models 34 i 17 (PFA)

A la Figura 20 mostrem els *colormaps* per a dos PFAs. En el de l'esquerra, del model n. 34, $(n = 64, |\Sigma| = 21)$, per exemple, es veu de manera força clara que s'haurien pogut aconseguir millors resultats prenent una base més gran. Això és consistent amb el fet que la mida de la base considerada òptima en la validació creuada ha estat la màxima, $10 \cdot |\Sigma| = 210$. A més a més, és un dels models que ha obtingut pitjor WER en relació amb el model real (diferència de 8.3 punts percentuals).

Aquesta circumstància, en què la mida de base òptima ha estat la màxima considerada en l'experiment, es repeteix per un gran nombre de models, la qual cosa té sentit ja que una quantitat més gran d'informació durà a un aprenentatge més acurat. Per a PFAs i DPFAs és on l'evidència que una base més gran permetria millorar substancialment el resultat és més clara: el 87% dels PFA i el 81% dels DPFA considerats han determinat òptima una base de mida $\geq 9 \cdot |\Sigma|$, per un 63% en el cas dels HMM.

Si observem les Figures 20 i 21, sembla que els PFA i els DPFA tenen un comportament bastant semblant, que podem veure una lleugerament diferenciat del mostrat a la Figura 22 (HMM). El ventall que s'obre al voltant dels valors òptims és veu força més ample per a PFAs i DPFAs, la qual cosa denota una menor sensibilitat a la variació dels paràmetres, mentre que el ventall obert al voltant dels valors òptims dels HMM és bastant més estret.

Concretament, la gràfica de la dreta de la Figura 22, corresponent al model n. 20 $(n = 11, |\Sigma| = 18)$, denota una sensibilitat molt alta a la variació en el nombre d'estats respecte del valor òptim $(n = 7)$,

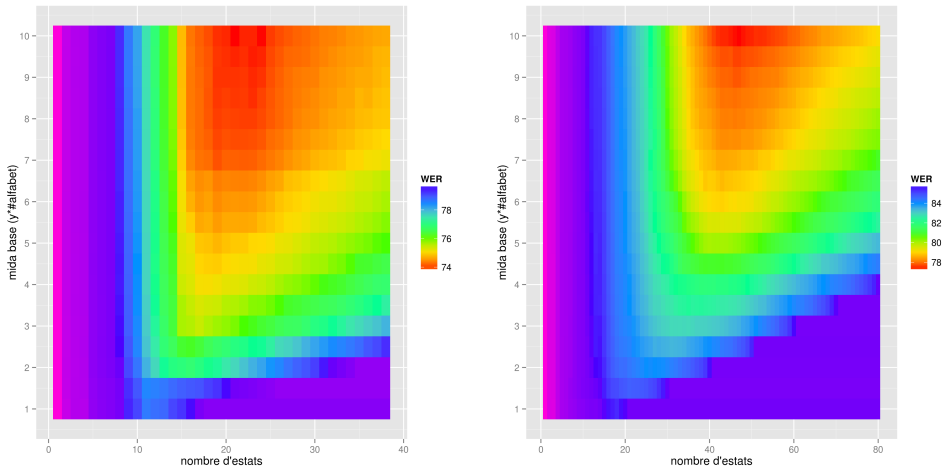


Figura 21: Sensibilitat a paràmetres per als models 27 i 11 (DPFA)

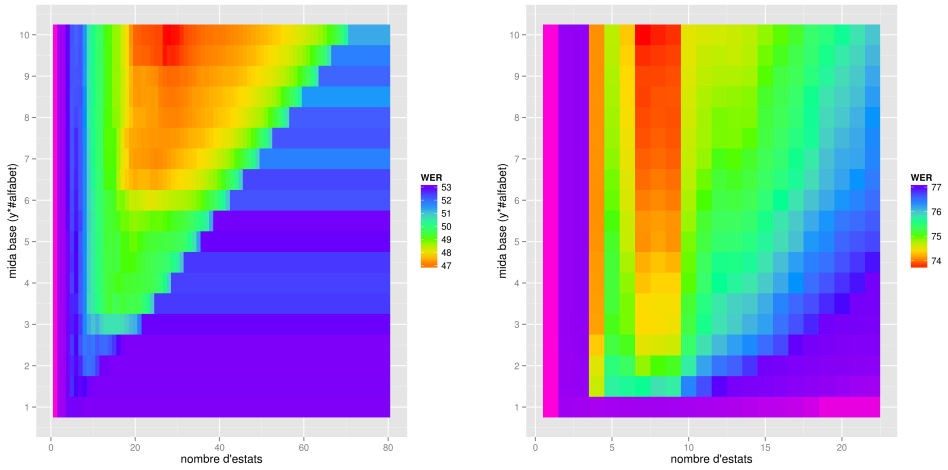


Figura 22: Sensibilitat a paràmetres per als models 19 i 20 (HMM)

tant a la baixa com a l'alça. La sensibilitat de la gràfica de l'esquerra també és molt alta, tot i que no s'aprecia tan bé degut a l'escala més àmplia.

Sembla que, en general, els models apresos mitjançant el mètode espectral seran molt més sensibles a variacions en el nombre d'estats que a variacions en la mida de la base. Les variacions en el nombre d'estats afecten negativament tant si són a la baixa com si són a l'alça, tot i que són més notables quan són a la baixa.

Per últim, a la Figura 23 hem volgut mostrar i comentar dos casos molt diferents a la resta. La gràfica de l'esquerra correspon al model n. 18 (DPFA, $n = 25, |\Sigma| = 20$), que a partir del nombre d'estats òptim ($n = 23$), mostra una variació gairebé imperceptible si el nombre d'estats és més alt. Noteu que, tot i que l'escala de color és força àmplia, el color vermell gairebé no varia.

A la gràfica de la dreta de la Figura 23, corresponent al model no. 25 (HMM, $n = 40, |\Sigma| = 10$), observem una distribució bastant irregular del WER que contrasta amb la resta de models observats, en què la distribució està bastant centrada al voltant d'un valor o una franja de valors mínims.

6.6 Altres experiments

En aquest apartat exposem alguns experiments que no s'han realitzat en aquest projecte per no haver estat contemplats en la planificació temporal, però que un cop realitzat el conjunt d'experiments que

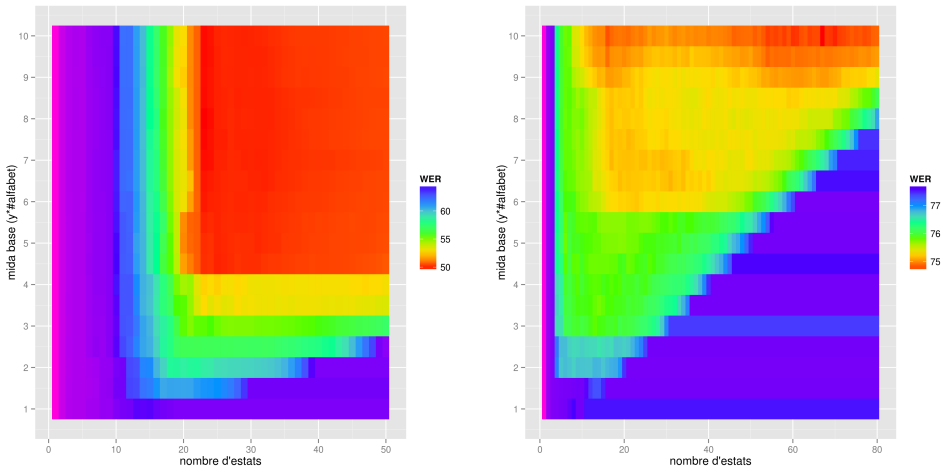


Figura 23: Sensibilitat a paràmetres per als models 18 i 25

s'ha descrit aquest capítol hem considerat interessants d'executar.

Noteu que cadascun dels experiments descrits a continuació tenen un cost elevat quant a temps de computació.

Corba d'aprenentatge En l'experiment de la Secció 6.4 s'han estudiat les corbes d'aprenentatge per a uns valors fixats del nombre d'estats i la mida de la base, obtinguts de l'*cross-validation* de l'experiment descrit a la Secció 6.3.

Proposem estudiar les corbes d'aprenentatge dels models incorporant variacions en els aquests paràmetres i analitzar com es comporten en funció d'aquestes variacions.

Sensibilitat quant a la *perplexity* En l'experiment de la Secció 6.5 s'ha estudiat la sensibilitat tenint en compte el WER obtingut per a cadascun dels valors del nombre d'estats i mida de la base.

Proposem estudiar la sensibilitat en funció de la *perplexity* del model espectral envers el model real.

Estudi de casos patològics L'estudi d'alguns models concrets ha donat lloc a la troballa d'alguns casos patològics. És el cas de la gràfica de la dreta de la Figura 23 en la Secció 6.5, en què s'observa una distribució irregular del WER. Per a aquest cas en concret, proposem estudiar si aquesta irregularitat desapareix si s'utilitza un *train set* més gran.

Anàlisi amb bases més grans En l'experiment de la Secció 6.5 ja hem esmentat que la mida de base que s'ha pres com a òptima ha estat, en la majoria dels casos, la màxima que s'ha considerat.

Proposem considerar bases més grans i analitzar si aquest comportament es manté per a valors més grans de la mida de la base i es tracta només del temps d'execució que requereix el mètode espectral a mesura que la base creix o si, per contra, els valors del WER obtingut comencen a ser pitjors a partir d'algun punt, com ja passa amb el nombre d'estats.

7 Valoració final

A mode de conclusió, en aquest apartat repasso i comento a nivell personal l'assoliment de cadascun dels objectius marcats a l'inici del projecte.

Mètode espectral en R S'ha obtingut una implementació correcta del mètode espectral. S'ha obtingut el paquet *spectralLearning* en el llenguatge R, un llenguatge de programació completament desconegut per mi en el moment d'iniciar el projecte. S'han integrat en el mateix codi font parts redactades en R i parts redactades en C++, aprofitant alhora els avantatges de tots dos llenguatges.

Evidentment poden millorar-se alguns aspectes del paquet, com podria ser el fet que la majoria de les funcions no fan control d'errors sobre els paràmetres, un punt que s'ha de tenir en compte donada la fragilitat que presenta el llenguatge R en aquest sentit.

Flux de treball La interacció de l'usuari amb el paquet s'ha dividit en diversos nivells, permetent fer-ne ús tant a usuaris amb poc coneixement de *machine learning* com a usuaris avançats. En tot cas, és feina de l'usuari transformar la seva mostra al format de llistes de vectors d'enters que és capaç de tractar el paquet implementat. A partir d'aquí, *spectralLearning* permet a l'usuari escollir entre desentendre's del flux d'execució i obtenir directament l'autòmat o, fins i tot, l'avaluació; o bé participar activament en el flux d'execució seleccionant els paràmetres que més li convinguin segons el cas.

Eficiència La integració de codi C++ en el codi R permet al paquet oferir un salt qualitatiu enfront d'un paquet que hagués estat implementat únicament en R. S'ha treballat al màxim l'aspecte de l'eficiència, tot i que s'han deixat sense optimitzar algunes funcionalitats menys crítiques per qüestions de temps: en molts casos el temps invertit en optimitzar no paga la pena del temps d'execució guanyat. És el cas, per exemple, del càlcul de la matriu de Hankel: reduir els dos segons de mitjana que triga en calcular-se en els casos estudiats és possible però no s'ha considerat crític. Sí s'ha treballat a fons, en canvi, l'optimització del procés d'obtenció de bases (vegeu Secció 5.2.1).

Documentació S'ha escrit la documentació del paquet *spectralLearning* en anglès i en el format estàndard de R. Per tal que fos més completa, caldria redactar una *vignette* del paquet, un document molt comú en els paquets de R que descriu a alt nivell com s'ha d'usar en paquet. Aquesta *vignette* hauria de ser una mena de tutorial, amb una estructura molt similar a la Secció 5.1, en què es descriu el paquet.

Aplicació S'ha implementat una comparació del model obtingut mitjançant el mètode espectral amb el model de trigrames, i s'ha provat que el model espectral supera el de trigrames en la gran majoria dels casos.

Memòria autocontinguda S'ha redactat una memòria autocontinguda i comprensible per a qualsevol persona amb cultura científica, sense entrar al detall en molts dels aspectes. A canvi de renunciar al detall, s'ha obtingut una memòria relativament breu, concreta i clara.

S'ha aconseguit, doncs, un assoliment general dels objectius inicialment marcats, si bé encara quedaria tocar alguns aspectes per acabar d'arrodonir el projecte: no s'han explorat tècniques de millora de l'aprenentatge com l'*smoothing* o la normalització de variància del subespai, ni s'han mogut els fils necessaris perquè el paquet *spectralLearning* pugi al CRAN, el repositori de la comunitat de R.

La realització d'aquest projecte ha estat en general una experiència molt positiva, integrant molts dels coneixements obtinguts durant els estudis i permetent-me un tast del món de la recerca acadèmica. És una llàstima que, ara per ara, es tingui tant poc respecte per la Universitat des de les institucions públiques. De no ser així, hauria de plantejar-me molt seriosament seguir pel camí acadèmic.

8 Bibliografia

Mostrem les referències més rellevants emprades per desenvolupar el projecte dividides en dues llistes. La primera, de referències relacionades amb el mètode espectral i aspectes que se'n deriven. La segona, purament relacionada amb la programació i el desenvolupament del paquet.

No cal oblidar algunes referències intangibles, com són l'assistència a algunes xerrades organitzades per LARCA i a les *IV Jornadas de Usuarios de R* organitzades per la comunitat R-Hispano a Barcelona el novembre de 2012.

8.1 El mètode espectral

- B. Balle. *Spectral and combinatorial algorithms for learning finite state machines.* Tesi doctoral en computació, 2013.
- B. Balle, A. Quattoni i X. Carreras. *Local Loss Optimization in Operator Models: A New Insight into Spectral Learning.* ICML, 2012.
- F. M. Luque, A. Quattoni, B. Balle i X. Carreras. *Spectral Learning for Non-Deterministic Dependency Parsing.* EACL, 2012.
- S. Verwer, R. Eyraud, C. de la Higuera. *Results of the PAutomaC Probabilistic Automaton Learning Competition.* ICGI, 2012.
- S. Verwer, R. Eyraud, C. de la Higuera. *PAutomaC: a PFA/HMM Learning Competition.* JMLR: Workshop and Conference Proceedings.
- L. Trefethen i D. Bau. *Numerical Linear Algebra.*
- Referència general a http://www.wikipedia.org

8.2 El paquet *spectralLearning*

- Introducció a R i referència sobre el llenguatge R a http://cran.r-project.org/doc/manuals/
- P. Burns. *The R Inferno.* Manual de referència de R i bones pràctiques.
- Introducció a Rcpp a https://github.com/hadley/devtools/wiki/Rcpp
- Referència sobre el llenguatge R a http://www.cookbook-r.com
- Documentació de la llibreria Rcpp a http://www.rcpp.org
- Referència i consulta a http://stackoverflow.com
- Referència i consulta a les llistes r-help i r-devel. http://www.r-project.org/mail.html
- Referència per a la confecció de la memòria a http://en.wikibooks.org/wiki/LaTeX
- Referència general a http://www.wikipedia.org

A Annex: Documentació del paquet *spectralLearning*

Package 'spectralLearning'

June 2, 2013

Type Package

Title Spectral method for automata learning

Version 0.1

Date 2013-06-03

Author Albert Santiago - LARCA - LSI - UPC

Maintainer Albert Santiago <albertsb@lsi.upc.edu>

Description Spectral method for automata learning

License None (yet)

Depends Rcpp (>= 0.10.1), MASS

LinkingTo Rcpp

R topics documented:

`evaluate.by.perplexity` *Automata evaluation by perplexity*

Description

Evaluates an automaton by perplexity

Usage

```
evaluate.by.perplexity(aut, word.list, solution.scores)
```

Arguments

`aut`	An automaton, as returned from `spectral.automaton`, it is: a list containing the attributes a1, ainf, A in the corresponding format.
`word.list`	A test set, consisting in a list of integer vectors -the test words.
`solution.scores`	A numeric vector containing the scores given to the test set by the real -or another- model. The sum of all the values must be 1.

Details

Parameters word.list and solution.scores must have the same length. Full parameter error-check is not available in this version.

Value

A numeric vector containing, in the first position, the perplexity of the solution against itself, and in the second position, the perplexity of the candidate scores given by `aut`.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

`perplexity`, `spectral.automaton`

`evaluate.by.wer` *Automata evaluation by word error rate*

Description

Evaluates an automaton by word error rate

Usage

```
evaluate.by.wer(aut, word.list)
wer.as.prefix.automaton(word.list, aut)
```

Arguments

`aut`	An automaton, as returned from `spectral.automaton`, it is: a list containing the attributes a1, ainf, A in the corresponding format.
`word.list`	A test set, consisting in a list of integer vectors -the test words.

Details

Parameter error-check is not available in this version.

Value

The prediction word error rate of `aut` in `word.list`, taking `aut` as a prefix automaton, it is: the function computed by `aut` is prob(x*Sigma^*). This is the method to use to compute the word error rate for an automaton learnt by spectral method in prefixes learning mode.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

`wer.as.PFA`, `spectral.automaton`, `wer.trigram`

evaluateWords *Automaton evaluation of words*

Description

Evaluates a word list by an automaton.

Usage

```
evaluateWords(word.list, aut)
```

Arguments

word.list	A test set, consisting in a list of integer vectors -the test words.
aut	An automaton, as returned from spectral.automaton, it is: a list containing the attributes a1, ainf, A in the corresponding format.

Details

Parameter error-check is not available in this version.

Value

Returns a numeric vector: For each word w in word.list, evaluateWords computes the value aut.a1 * aut.A_w * aut.ainf.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

evaluate.by.perplexity, perplexity

getAutomaton *Automata construction for spectral method*

Description

Gets an automaton from the Hankel matrices for a sample.

Usage

```
getAutomaton(n, alphSize, H, Hsigma)
```

Arguments

n	Number of states of the automaton returned.
alphSize	Size of the alphabet.
H	Hankel matrix.
Hsigma	Hankel-sigma matrices: a list of alphSize matrices.

Details

Parameter error-check is not available in this version.

Value

Returns a an automaton learnt from the Hankel matrices in a list with attributes:

a1	The initial values for the automaton
ainf	The final values for the automaton
A	A list formed by the transition matrices for the automaton

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

spectral.automaton,getBase,getHankelMatrices

---

getBase *Basis construction*

Description

Gets a basis from a sample for the spectral method.

Usage

```
getBase(pref.size, suff.size, word.list, learning.type)
```

Arguments

pref.size	Size of the prefix set generated.
suff.size	Size of the suffix set generated.
word.list	The sample, consisting in a list of integer vectors -the words.
learning.type	Learning mode to use. It can be 1 (for strings), 0 (for prefixes) or 2 (for substrings).

Details

Parameter error-check is not available in this version.

Value

A basis, in a list:

p	Prefix set for the basis, consisting in a list of integer vectors -the prefixes
s	Suffix set for the basis, consisting in a list of integer vectors -the suffixes

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

spectral.automaton,getHankelMatrices,getAutomaton

---

getSampleFromModel *Sample construction from PAutomaC model*

Description

Constructs a sample from a model in PAutomaC format

Usage

```
getSampleFromModel(model, N)
```

Arguments

model	An unnamed list, with attributes: the initial values, the final values, the symbol generation probabilities and the transition probabilities
N	Desired size for the sample

Details

Parameter error-check is not available in this version.

Value

The generated sample, consisting in a list of integer vectors -the words.

Author(s)

Albert Santiago - LARCA - LSI - UPC

initialize.hankel.sigma *Matrix list initialization*

Description

Matrix list memory allocation

Usage

```
initialize.hankel.sigma(alph.size, base.pref.size, base.suff.size)
```

Arguments

alph.size	List length.
base.pref.size	Number of rows for each matrix.
base.suff.size	Number of columns for each matrix.

Value

Returns a list of length alph.size. Each element of the list is a matrix of base.pref.size rows and base.suff.size columns, initialized to 0.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

spectral.automaton,getBase,getHankelMatrices,getAutomaton

modelToAutomaton *PAutomaC models conversion*

Description

Converts a model in PAutomaC format to an automaton

Usage

```
modelToAutomaton(model)
```

Arguments

model	An unnamed list, with attributes: the initial values, the final values, the symbol generation probabilities and the transition probabilities

Details

Parameter error-check is not available in this version.

Value

Returns the converted automaton in a list with attributes:

a1	The initial values for the automaton
ainf	The final values for the automaton
A	A list formed by the transition matrices for the automaton

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

getSampleFromModel

---

perplexity *Perplexity between two probability distributions*

Description

Computes the perplexity between two probability distributions.

Usage

```
perplexity(cand, sol)
```

Arguments

cand	Probability distribution for a test set computed by a candidate automaton.
sol	Correct probability distribution for a test set.

Details

cand and sol must be numeric vectors of the same length. sum(cand) and sum(sol) both have to be 1.

Value

A single numeric value containing the perplexity of distribution cand against sol.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

`evaluate.by.perplexity`, `evaluateWords`

`print.word` *Friendly printing*

Description

Prints a word -in the form of an integer vector- in a more readable format, given a the correspondency between numbers and symbols

Usage

```
print.word(alph, word, sym.sep = "")
```

Arguments

`alph`	A character vector
`word`	An integer vector
`sym.sep`	A string unique value

Details

Parameter error-check is not available in this version.

Value

Returns a single string, obtained by concatenating the translated symbols of `word` separated by `sym.sep`.

Author(s)

Albert Santiago - LARCA - LSI - UPC

`spectral.automaton` *Automata learning by spectral method*

Description

Learns an automaton from a sample using spectral method

Usage

```
spectral.automaton(word.list, alph.size, n.states = "default", base.pref.size = "default", base.suff.s
```

Arguments

`word.list`	The sample, consisting in a list of integer vectors -the words.
`alph.size`	Size of the alphabet.
`n.states`	Number of states of the automaton returned. Default value is allowed.
`base.pref.size`	Size of the prefix set generated to compute the Hankel matrix. Default value is allowed.
`base.suff.size`	Size of the suffix set generated to compute the Hankel matrix. Default value is allowed.
`learning.mode`	Learning mode to use. It can be "strings", "prefixes" or "substrings". Default value is "prefixes".

Details

Partial parameter error-check is performed. Not checking if all word symbols are in the alphabet.

Value

Returns an automaton learnt from the sample in a list with attributes:

`a1`	The initial values for the automaton
`ainf`	The final values for the automaton
`A`	A list formed by the transition matrices for the automaton

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

`getBase`,`getHankelMatrices`,`getAutomaton`

`substring.2.prefix.aut` *Automata conversion*

Description

Converts an automaton that predicts over substrings to the equivalent automaton that predicts over prefixes or strings.

Usage

```
substring.2.prefix.aut(aut)
substring.2.string.aut(aut)
```

Arguments

`aut`	An automaton, as returned from `spectral.automaton`, it is: a list containing the attributes a1, ainf, A in the corresponding format.

Details

Parameter error-check is not available in this version.

Value

A list, containing an automaton that predicts over prefixes or strings, obtained by doing the appropiate conversion to the parameter `aut`. The list has the following attributes:

`a1`	The initial values for the automaton
`ainf`	The final values for the automaton
`A`	A list formed by the transition matrices for the automaton

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

`spectral.automaton`

`trigram.model` *n-gram language learning*

Description

Gets a n-gram model from a sample.

Usage

```
unigram.model(word.list)
bigram.model(word.list)
trigram.model(word.list)
```

Arguments

`word.list`	The sample, consisting in a list of integer vectors -the words.

Details

Parameter error-check is not available in this version.

Value

A unique integer value is returned for unigram.model function, containing the symbol to predict by the unigram model. bigram.model and trigram.model return a list containing:

`unigram`	An integer
`bigram`	An integer vector, containing the symbol to predict after each symbol of the alphabet
`trigram`	(only trigram.model) An integer array, containing the symbol to predict after each pair of symbols

NA values are used when there is not enought information in the sample.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

`wer.trigram`

wer.as.PFA — *Automata evaluation by word error rate*

Description

Evaluates an automaton by word error rate

Usage

```
wer.as.PFA(word.list, aut)
```

Arguments

word.list	A test set, consisting in a list of integer vectors -the test words.
aut	An automaton, as returned from spectral.automaton, it is: a list containing the attributes a1, ainf, A in the corresponding format.

Details

Parameter error-check is not available in this version.

Value

The prediction word error rate of aut in word.list, taking aut as a PFA, it is: the function computed by aut is prob(x) and not prob(x*Sigma^*) as in prefix automata.

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

evaluate.by.wer, wer.trigram

wer.trigram — *n-gram evaluation by word error rate*

Description

Evaluates a n-gram by word error rate

Usage

```
wer.trigram(word.list, n.gram)
wer.bigram(word.list, n.gram)
wer.unigram(word.list, unigram)
```

Arguments

word.list	A test set, consisting in a list of integer vectors -the test words.
n.gram	A n-gram, as returned from trigram.model or bigram.model, it is: a list containing the atributes unigram, bigram and trigram, if necessary.
unigram	An unigram, as returned from unigram.model, it is: a single integer value.

Details

Parameter error-check is not available in this version.

Value

The prediction word error rate of n.gram or unigram in word.list

Author(s)

Albert Santiago - LARCA - LSI - UPC

See Also

trigram.model

Index